Virtuozzo 7

Virtualization SDK: C API Reference

March 23, 2017

Parallels International GmbH

Vordergasse 59

8200 Schaffhausen

Switzerland

Tel: + 41 52 632 0411

Fax: + 41 52 672 2010

http://www.virtuozzo.com

Copyright ©1999-2017 Parallels International GmbH. All rights reserved.

This product is protected by United States and international copyright laws. The product's underlying technology, patents, and trademarks are listed at https://virtuozzo.com.

Microsoft, Windows, Windows Server, Windows NT, Windows Vista, and MS-DOS are registered trademarks of Microsoft Corporation.

Apple, Mac, the Mac logo, Mac OS, iPad, iPhone, iPod touch, FaceTime HD camera and iSight are trademarks of Apple Inc., registered in the US and other countries.

Linux is a registered trademark of Linus Torvalds. All other marks and names mentioned herein may be trademarks of their respective owners.

Table of Contents

Introduction 1

About This Guide 1

How to Use This Guide 2

Organization of This Guide 2

Functions by Handle 4

PHT_ACCESS_RIGHTS 9

PHT_BACKUP 13

PHT_BACKUP_RESULT 13

PHT_BOOT_DEVICE 14

PHT_CPU_FEATURES 22

PHT_CPU_POOL 24

PHT_CT_TEMPLATE 26

PHT_CVSRC 32

PHT_DESKTOP_CONTROL 32

PHT_DISP_CONFIG 32

PHT_EVENT 72

PHT_EVENT_PARAMETER 88

PHT_FIREWALL_RULE 95

PHT_FOUND_VM_INFO 102

PHT_GUEST_OSES_MATRIX 105

PHT_HANDLES_LIST 108

PHT_HW_GENERIC_DEVICE 111

PHT_HW_GENERIC_PCI_DEVICE 115

PHT_HW_HARD_DISK 117

PHT_HW_HARD_DISK_PARTITION 121

PHT_HW_NET_ADAPTER **126**

PHT_IPPRIV_NET **133**

PHT_JOB **138**

PHT_LICENSE **146**

PHT_LOGIN_RESPONSE **149**

PHT_NET_SERVICE_STATUS **153**

PHT_NETWORK_CLASS **154**

PHT_NETWORK_RATE **157**

PHT_NETWORK_SHAPING **160**

PHT_NETWORK_SHAPING_BANDWIDTH **166**

PHT_NETWORK_SHAPING_CONFIG **169**

PHT_OFFLINE_SERVICE **173**

PHT_OPAQUE_TYPE_LIST **177**

PHT_PLUGIN_INFO **180**

PHT_PORT_FORWARDING **185**

PHT_PROBLEM_REPORT **192**

PHT_REMOTE_FILESYSTEM_ENTRY **202**

PHT_REMOTE_FILESYSTEM_INFO **206**

PHT_RESULT **210**

PHT_RUNNING_TASK **217**

PHT_SCREEN_RESOLUTION **219**

PHT_SERVER **219**

PHT_SERVER_CONFIG **314**

PHT_SERVER_INFO **344**

PHT_SESSION_INFO **350**

PHT_SHARE **350**

PHT_STRINGS_LIST **357**

PHT_SYSTEM_STATISTICS **360**

PHT_SYSTEM_STATISTICS_CPU **365**

PHT_SYSTEM_STATISTICS_DISK **367**

PHT_SYSTEM_STATISTICS_DISK_PARTITION **367**

PHT_SYSTEM_STATISTICS_IFACE **368**

PHT_SYSTEM_STATISTICS_PROCESS **368**

PHT_SYSTEM_STATISTICS_USER_SESSION **368**

PHT_SYSTEM_STATISTICS_VM_DATA **368**

PHT_UIEMU_INPUT **368**

PHT_USB_IDENTITY **368**

PHT_USER_INFO **371**

PHT_USER_PROFILE **374**

PHT_VCMMD_CONFIG **379**

PHT_VIDEO_RECEIVER **381**

PHT_VIRTUAL_DEV_FLOPPY **381**

PHT_VIRTUAL_DEV_HARD_DISK **381**

PHT_VIRTUAL_DEV_HD_PARTITION **397**

PHT_VIRTUAL_DEV_NET_ADAPTER **399**

PHT_VIRTUAL_DEV_OPTICAL_DISK **425**

PHT_VIRTUAL_DEV_PARALLEL_PORT **425**

PHT_VIRTUAL_DEV_SERIAL_PORT **425**

PHT_VIRTUAL_DEV_SOUND **427**

PHT_VIRTUAL_DEV_USB_DEVICE **430**

PHT_VIRTUAL_DISK **432**

PrlDiskOpenPolicy_CreateOffset Function 440

PHT_VIRTUAL_DISK_ENCRYPTION **440**

PHT_VIRTUAL_MACHINE **442**

PHT_VIRTUAL_NETWORK **507**

PHT_VM_CONFIGURATION **537**

PHT_VM_GUEST_SESSION **716**

PHT_VM_INFO 719

PHT_VM_TOOLS_INFO 724

PHT_VM_VIRTUAL_DEVICES_INFO 726

Common Functions 727

System Functions 727

API Debug Functions 744

Handle Functions 747

Virtual Device Functions 752

PrlBuffer 775

Remote Desktop Access 777

Audio Playback 778

Keyboard Control 778

Mouse Control 778

Primary Display Capture 778

Secondary Display Capture 778

Callback Prototypes 779

PRL_DISK_CALLBACK Type 779

PRL_EVENT_HANDLER_PTR 780

PRL_STATES_CALLBACK Type 780

Types 782

PRL_BOOL Types 782

PRL_CHAR Types 783

PRL_CONST Types 784

PRL_DEVICE Types 788

PRL_DISK Types 788

PRL_DOUBLE Types 789

PRL_FILE Types 790

PRL_INT Types 791

PRL_IO Types 794

PRL_TIS Types 796

PRL_TOOLS Types 796

PRL_UINT Types 797

PRL_VM Types 800

PRL_VOID Types 808

Other PRL Types 809

Other Types 835

Structs, Records, Enums 837

PRL_IO Structs 837

PRL_VM Structs 920

Other PRL Structs 953

Other Structs 1102

Macros 1106

PET Macros 1106

PRL_DISK Macros 1128

PRL_ERR Macros 1137

PRL_GUEST Macros 1499

PRL_IMAGE Macros 1503

PRL_NET Macros 1508

PRL_QUEST Macros 1521

PRL_QUESTION Macros 1523

PRL_SCAN_CODE Macros 1526

PRL_STATE Macros 1561

PRL_WARN Macros 1564

PRL_WRN Macros **1575**

PVS_GUEST Macros **1576**

Other PRL Macros **1620**

Other Macros **1654**

Index f

Introduction

Topics

Name	Description
About This Guide (p. 1)	This document is a part of the Virtuozzo Virtualization SDK. It contains the Virtuozzo C API reference including information about handles (objects), functions, types, structures, enumerations, and other technical information. Please note that this document does not provide general information on how to use the Virtuozzo Virtualization SDK. For the detailed explanation of how to start using the SDK and how to create your own programs using the provided APIs, please read the **Virtuozzo Virtualization SDK Programmer's Guide**, which is a companion book to this one. To understand the material provided in this guide, you should have a basic... more (p. 1)
How to Use This Guide (p. 2)	As a demonstration, let's select a task and see how you can use this guide to find the technical information that will help you to implement it. We will use the task of starting a virtual machine. Follow these steps: 1. Everything in the Virtuozzo C API begins with a *handle*. If you want to perform a task, you need to know the name of the handle that provides the corresponding functionality (you can browse the **Functions by Handle (p. 4)** chapter and read the summary for each handle). In our case, the name of the handle is `PHT_VIRTUAL_MACHINE ( p. 442)`. 2. Read the handle description and... more (p. 2)
Organization of This Guide (p. 2)	This guide is organized into the following chapters: **Functions by Handle**. Contains sections describing handles, each of which contains a list of functions that can operate on that handle. Each section contains information about the purpose of a handle and how to obtain a reference to it. Function subsections contain the usual reference information such as syntax, parameters, return codes, and remarks. You can search the guide for a specific function or a keyword or you can scroll through the handle list, find the handle providing the functionality of interest, and then look for a specific function in its... more (p. 2)

About This Guide

This document is a part of the Virtuozzo Virtualization SDK. It contains the Virtuozzo C API reference including information about handles (objects), functions, types, structures, enumerations, and other technical information.

Please note that this document does not provide general information on how to use the Virtuozzo Virtualization SDK. For the detailed explanation of how to start using the SDK and how to create your own programs using the provided APIs, please read the **Virtuozzo Virtualization SDK Programmer's Guide**, which is a companion book to this one.

To understand the material provided in this guide, you should have a basic knowledge of the ANSI C programming language and some experience working with application programming interfaces (API) written in C. You should also be familiar with Virtuozzo virtualization products.

How to Use This Guide

As a demonstration, let's select a task and see how you can use this guide to find the technical information that will help you to implement it. We will use the task of starting a virtual machine. Follow these steps:

1. Everything in the Virtuozzo C API begins with a *handle*. If you want to perform a task, you need to know the name of the handle that provides the corresponding functionality (you can browse the **Functions by Handle (p. 4)** chapter and read the summary for each handle). In our case, the name of the handle is `PHT_VIRTUAL_MACHINE ( p. 442)`.
2. Read the handle description and find the information about how to obtain it. In this case, the handle is obtained from the `PHT_JOB ( p. 138)` object returned by the `PrlSrv_GetVmList ( p. 264)` function. Follow the link to the `PrlSrv_GetVmList ( p. 264)` function.
3. When the topic describing the `PrlSrv_GetVmList ( p. 264)` function is displayed, you will see a detailed description of the function and a sample demonstrating how to use it. The function accepts a handle of type `PHT_SERVER ( p. 219)` as parameter. Follow its link to see how it is obtained (or see the included code sample). The `PHT_JOB ( p. 138)` object that the PrlSrv_GetVmList (p. 264) function returns will contain the `PHT_VIRTUAL_MACHINE ( p. 442)` handle that we want. The information on how to extract the handle from the Job object is included in the function description. You can also follow the `PHT_JOB ( p. 138)` link and see more info about it if you wish.
4. Now that you know how to obtain the `PHT_VIRTUAL_MACHINE ( p. 442)` handle, go back to its topic (use the document viewer options or click the link if it's available on the current page).
5. In the `PHT_VIRTUAL_MACHINE ( p. 442)` function list, find the `PrlVm_Start ( p. 491)` function (functions are listed alphabetically). This is the function that starts a virtual machine. Follow its link and see the syntax, parameters, and return codes. As you can see, the function accepts a single parameter, which is a handle of type `PHT_VIRTUAL_MACHINE ( p. 442)` that we've obtained in the previous steps. You now have all the information needed to implement the task in your program.

Once again, if you are just starting with the Virtuozzo Virtualization SDK, your best source of information is **Virtuozzo Virtualization SDK Programmer's Guide**. It explain the concepts and provides descriptions and step-by-step instructions on how to use the API to perform the most common tasks. Once you are familiar with the basic principles of the SDK design, you will find it easy to work with both guides and the API itself.

Organization of This Guide

This guide is organized into the following chapters:

Functions by Handle. Contains sections describing handles, each of which contains a list of functions that can operate on that handle. Each section contains information about the purpose of a handle and how to obtain a reference to it. Function subsections contain the usual reference information such as syntax, parameters, return codes, and remarks. You can search the guide for a specific function or a keyword or you can scroll through the handle list, find the handle providing the functionality of interest, and then look for a specific function in its function list.

Common Functions. Contains information about common and system functions.

Remote Desktop Access. Contains information about functions that can be used to remotely capture screenshots of a virtual machine desktop and to send keyboard and mouse commands to it.

Callback Prototypes. Describes callback function prototypes that are used to implement your own callback functions for use in asynchronous calls.

Types. Contains a list of types used throughout the API.

Structs, Records, Enums. Contains information about structures, records, and enumerations used throughout the API.

Macros. Describes macros used in the API.

Files. Contains a list of the Virtuozzo C API header files. You can look at individual files in this chapter and see the functions, types (etc.) that they contain.

Functions by Handle

This chapter contains a list of handles. Each section describes a handle and contains subsections describing functions that can operate on the handle. Handle types are defined in the PRL_HANDLE_TYPE (p. 1064) enumeration.

Topics

Name	Description
PHT_ACCESS_RIGHTS (p. 9)	The PHT_ACCESS_RIGHTS handle is used to obtain and modify the virtual machine access rights information.
PHT_BACKUP (p. 13)	
PHT_BACKUP_RESULT (p. 13)	The PHT_BACKUP_RESULT handle is a container that contains the results of a backup operation. The handle is returned by the PrlSrv_CreateVmBackup (p. 233) function call (the handle is returned as the first parameter of the Job object).
PHT_BOOT_DEVICE (p. 14)	The PHT_BOOT_DEVICE handle is used to manage boot devices in a virtual machine. The handle is obtained by calling the PrlVmCfg_GetBootDev (p. 558) or the PrlVmCfg_CreateBootDev (p. 550) function. The first function is used to obtain a handle to an existing device in the boot priority list. The second function creates a new handle and is used to add a new device to the list.
PHT_CPU_FEATURES (p. 22)	
PHT_CPU_POOL (p. 24)	
PHT_CT_TEMPLATE (p. 26)	The PHT_CT_TEMPLATE handle is used to manage operating system templates on which Containers are based.
PHT_CVSRC (p. 32)	The PHT_CVSRC handle is used to capture data from a video source on the host computer and display in the guest operating system.
PHT_DESKTOP_CONTROL (p. 32)	The PHT_DESKTOP_CONTROL handle is used to remotely access the desktop of the physical host computer. Use the PrlSrv_CreateDesktopControl to create the handle. See the **Remote Desktop Access** section for the list of functions that support this functionality.
PHT_DISP_CONFIG (p. 32)	The PHT_DISP_CONFIG handle is used to manage Virtuozzo Service preferences. The handle is obtained by calling the PrlSrv_GetCommonPrefs (p. 244) function.
PHT_EVENT (p. 72)	The PHT_EVENT handle is a container that can contain information about system events or extended error information in asynchronous functions.
PHT_EVENT_PARAMETER (p. 88)	The PHT_EVENT_PARAMETER handle is a container that contains system event parameter data. The handle is obtained by calling the PrlEvent_GetParam (p. 84) function.
PHT_FIREWALL_RULE (p. 95)	The PHT_FIREWALL_RULE handle is used to create and modify firewall rules.
PHT_FOUND_VM_INFO (p. 102)	The PHT_FOUND_VM_INFO handle is a container that contains information about a virtual machine as a result of a search operation. The handle is obtained by calling the PrlSrv_StartSearchVms (p. 299) function.
PHT_GUEST_OSES_MATRIX (p. 105)	The PHT_GUEST_OSES_MATRIX handle is a container that contains a list of operating system names and versions that are supported on a given server. To obtain the handle, use the PrlSrv_GetSupportedOses (p. 259) function.

PHT_HANDLES_LIST (p. 108)	The PHT_HANDLES_LIST handle is a generic container that can hold a list of other handles. It is typically used to pass a list of handles to a function in a single parameter. The handles in the container can be of the same type or of different types. A new handle is created by making the PrlApi_CreateHandlesList (p. 728) function call.
PHT_HW_GENERIC_DEVICE (p. 111)	PHT_HW_GENERIC_DEVICE is a base handle used to manage physical devices. A physical device can be a real physical device installed on the host computer or, in some cases, a device in a virtual machine as seen by the guest operating system (this handle should not be confused with handles that are used to manage virtual devices).
PHT_HW_GENERIC_PCI_DEVICE (p. 115)	The PHT_HW_GENERIC_PCI_DEVICE handle is used to obtain information about a generic PCI device installed in the host machine. The handle is obtained by calling the PrlSrvCfg_GetGenericPciDevice (p. 323) function. The handle is derived from PHT_HW_GENERIC_DEVICE (p. 111), which provides additional functions.
PHT_HW_HARD_DISK (p. 117)	The PHT_HW_HARD_DISK handle is used to manage physical hard disks installed on the host computer. The handle is obtained by calling the PrlSrvCfg_GetHardDisk (p. 326) function. The handle is derived from PHT_HW_GENERIC_DEVICE (p. 111), which provides additional functions.
PHT_HW_HARD_DISK_PARTITION (p. 121)	The PHT_HW_HARD_DISK_PARTITION handle is used to manage partitions of a physical hard disk installed on the host computer. The handle is obtained by calling the PrlSrvCfgHdd_GetPart (p. 119) function. The handle is derived from PHT_HW_GENERIC_DEVICEt, which provides additional functions.
PHT_HW_NET_ADAPTER (p. 126)	The PHT_HW_NET_ADAPTER handle is used to manage network adapters installed on the host computer or in a guest operating system. The handle is obtained by calling the PrlSrvCfg_GetNetAdapter (p. 332) function. The handle is derived from the PHT_HW_GENERIC_DEVICE (p. 111) handle, which provides additional functions.
PHT_IPPRIV_NET (p. 133)	The PHT_IPPRIV_NET handle is used to create and configure IP private networks.
PHT_JOB (p. 138)	The PHT_JOB handle provides methods for managing asynchronous operations.
PHT_LICENSE (p. 146)	The PHT_LICENSE handle provides methods for retrieving the Virtuozzo license information. The handle is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetLicenseInfo (p. 249) function.
PHT_LOGIN_RESPONSE (p. 149)	The PHT_LOGIN_RESPONSE handle is a container that contains information pertinent to a successful Virtuozzo Service login operation. The handle is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_Login (p. 272) or the PrlSrv_LoginLocal (p. 276) function.
PHT_NET_SERVICE_STATUS (p. 153)	The PHT_NET_SERVICE_STATUS handle provides methods for retrieving the Virtuozzo Service network status information. A handle to it is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetNetServiceStatus (p. 249) function.
PHT_NETWORK_CLASS (p. 154)	The PHT_NETWORK_CLASS handle is used to configure network accounting. To use the handle, first create it using the PrlNetworkClass_Create (p. 155) function and then use the handle function to manage network classes configuration.
PHT_NETWORK_RATE (p. 157)	The PHT_NETWORK_RATE handle is used to configure the virtual machine output bandwidth. The handle is obtained using the PrlNetworkRate_Create (p. 158) function.
PHT_NETWORK_SHAPING (p. 160)	The PHT_NETWORK_SHAPING handle is used to configure network shaping. The handle is obtained using the PrlNetworkShapingEntry_Create (p. 160) function.

PHT_NETWORK_SHAPING_BANDWIDTH (p. 166)	The PHT_NETWORK_SHAPING_BANDWIDTH handle is used to create network shaping configuration.
PHT_NETWORK_SHAPING_CONFIG (p. 169)	The PHT_NETWORK_SHAPING_CONFIG handle is used to configure network shaping.
PHT_OFFLINE_SERVICE (p. 173)	The PHT_OFFLINE_SERVICE handle is used to configure offline services. When creating a new offline service configuration, use the PrlOffmgmtService_Create (p. 173) to create the handle. When managing existing offline services configurations, use the PrlSrv_GetOfflineServicesList (p. 252) function to obtain the list of handles.
PHT_OPAQUE_TYPE_LIST (p. 177)	The PHT_OPAQUE_TYPE_LIST handle is a container for storing type lists.
PHT_PLUGIN_INFO (p. 180)	The PHT_PLUGIN_INFO handle is a container that contains information about a third-party plug-in installed in Virtuozzo Service. To obtain the list of installed plug-ins use the PrlSrv_GetPluginsList (p. 254) function.
PHT_PORT_FORWARDING (p. 185)	The PHT_PORT_FORWARDING handle provides access to the Port Forwarding functionality. Using this functionality, you can redirect all incoming data from a specific TCP or UDP port in the host machine to the specified port in a virtual machine.
PHT_PROBLEM_REPORT (p. 192)	The PHT_PROBLEM_REPORT handle is used to obtain and manage Virtuozzo Problem Reports. To obtain the handle, use the PrlApi_CreateProblemReport (p. 729) function.
PHT_REMOTE_FILESYSTEM_ENTRY (p. 202)	The PHT_REMOTE_FILESYSTEM_ENTRY handle, together with the PHT_REMOTE_FILESYSTEM_INFO (p. 206) handle, is used to obtain information about the file system on the host machine.The PHT_REMOTE_FILESYSTEM_ENTRY handle is a container that contains information about a single file system entry. The handle is obtained by making the PrlFsInfo_GetChildEntry (p. 207) call.
PHT_REMOTE_FILESYSTEM_INFO (p. 206)	The PHT_REMOTE_FILESYSTEM_INFO handle is a container that contains information about a file system entry and its immediate child elements (e.g. directory/files). The handle is obtained from a PHT_JOB (p. 138) object returned by the PrlSrv_FsGetDirEntries (p. 239) or the PrlSrv_FsGetDiskList (p. 240) function.
PHT_RESULT (p. 210)	The PHT_RESULT handle is a container that contains results of an asynchronous operation. The handle is obtained from the PHT_JOB (p. 138) object returned by the asynchronous function.
PHT_RUNNING_TASK (p. 217)	The PHT_RUNNING_TASK handle is used to recover from the lost Virtuozzo Service connection. It allows to attach to an existing task that was started in the previous session and is still running inside the Virtuozzo Service. The handle is obtained using the PrlLoginResponse_GetRunningTaskByIndex (p. 151) function. Only a limited number of tasks is supported by this functionality. They are defined in the PRL_RUNNING_TASK_TYPE_ENUM (p. 1090) enumeration. For more information, please also see the PrlSrv_AttachToLostTask (p. 224) function.
PHT_SCREEN_RESOLUTION (p. 219)	The PHT_SCREEN_RESOLUTION handle is used to modify the list of screen resolutions available in a virtual machine. By default, only the most common resolutions are supported in a virtual machine. Using this handle, you can add additional resolutions and modify the existing ones if needed. The handle is obtained using the PrlVm_CreateScrRes or the PrlVm_GetScrRes function. The additional screen resolution support can be enabled or disabled in a virtual machine. To determine the status of the feature, use the PrlVmCfg_IsScrResEnabled (p. 636) function.
PHT_SERVER (p. 219)	At the root of the Virtuozzo API class hierarchy is a handle of type PHT_SERVER. The handle can be obtained using the PrlSrv_Create (p. 230) function. The PHT_SERVER handle is required to use most of the functionality within the Virtuozzo API.

PHT_SERVER_CONFIG (p. 314)	The PHT_SERVER_CONFIG handle is used to obtain the host computer configuration information. The handle is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetSrvConfig (p. 257) function call. The handle can also be used to obtain network settings of a guest operating system installed in a virtual machine. In this case, the handle is obtained using the PrlVmGuest_GetNetworkSettings (p. 716) function.
PHT_SERVER_INFO (p. 344)	The PHT_SERVER_INFO handle is a container that contains information about a host machine. The information includes the Virtuozzo Service port number, host name, OS version, product version, and host machine UUID. To obtain this information for the specified Virtuozzo Service, use the PrlSrv_GetServerInfo (p. 256) function. You can also search for Virtuozzo Services running on your network by executing the PrlSrv_LookupParallelsServers (p. 279) function. The function returns a list of PHT_SERVER_INFO handles for the Virtuozzo Services that it finds. The returned information can be used to establish a connection with the Virtuozzo Service of interest.
PHT_SESSION_INFO (p. 350)	
PHT_SHARE (p. 350)	The PHT_SHARE handle is used to manage host shares. With this handle, you can make a host OS folder visible and accessible in a virtual machine. The handle is obtained by making the PrlVmCfg_GetShare (p. 603) or the PrlVmCfg_CreateShare (p. 550) function call.
PHT_STRINGS_LIST (p. 357)	The PHT_STRING_LIST handle is a generic string container. A new handle is created by making the PrlApi_CreateStringsList (p. 730) function call.
PHT_SYSTEM_STATISTICS (p. 360)	The PHT_SYSTEM_STATISTICS handle is used to obtain performance statistics for the host computer or an individual virtual machine.
PHT_SYSTEM_STATISTICS_CPU (p. 365)	The PHT_SYSTEM_STATISTICS_CPU handle is used to obtain CPU statistics. The handle is obtained by making the PrlStat_GetCpuStat (p. 362) function call.
PHT_SYSTEM_STATISTICS_DISK (p. 367)	The PHT_SYSTEM_STATISTICS_DISK object is used to obtain disk statistics. The handle is obtained by making the PrlStat_GetDiskStat function call.
PHT_SYSTEM_STATISTICS_DISK_PARTITION (p. 367)	The PHT_SYSTEM_STATISTICS_DISK_PARTITION handle is used to obtain disk partition statistics. The handle is obtained by making the PrlStatDisk_GetPartStat function call.
PHT_SYSTEM_STATISTICS_IFACE (p. 368)	The PHT_SYSTEM_STATISTICS_IFACE handle is used to obtain network interface statistics. The handle is obtained by making the PrlStat_GetIfaceStat function call.
PHT_SYSTEM_STATISTICS_PROCESS (p. 368)	The PHT_SYSTEM_STATISTICS_PROCESS handle is used to obtain system process statistics. The handle is obtained by making the PrlStat_GetProcStat function call.
PHT_SYSTEM_STATISTICS_USER_SESSION (p. 368)	The PHT_SYSTEM_STATISTICS_USER_SESSION handle is used to obtain user session statistics. The handle is obtained by making the PrlStat_GetUserStat function call.
PHT_SYSTEM_STATISTICS_VM_DATA (p. 368)	
PHT_UIEMU_INPUT (p. 368)	
PHT_USB_IDENTITY (p. 368)	
PHT_USER_INFO (p. 371)	The PHT_USER_INFO handle is a container that contains information about an individual user. The handle is obtained by making the PrlSrv_GetUserInfo (p. 260) or the PrlSrv_GetUserInfoList (p. 261) function call. The handle can be obtained for any user (not just the currently logged in user). To obtain the information for the currently logged in user, see the PHT_USER_PROFILE (p. 374) handle.

PHT_USER_PROFILE (p. 374)	The PHT_USER_PROFILE handle is used to obtain information about the currently logged in user and for setting the user preferences. The handle is obtained by calling the PrlSrv_GetUserProfile (p. 262) function. To obtain information about any user (not just the currently logged on user), see the PHT_USER_INFO (p. 371) handle.
PHT_VCMMD_CONFIG (p. 379)	
PHT_VIDEO_RECEIVER (p. 381)	
PHT_VIRTUAL_DEV_FLOPPY (p. 381)	The PHT_VIRTUAL_DEV_FLOPPY handle contains information about a virtual floppy disk in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetFloppyDisk (p. 575) function call.
PHT_VIRTUAL_DEV_HARD_DISK (p. 381)	The PHT_VIRTUAL_DEV_HARD_DISK handle is used to manage virtual hard disks in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetHardDisk (p. 580) function call.
PHT_VIRTUAL_DEV_HD_PARTITION (p. 397)	The PHT_VIRTUAL_DEV_HD_PARTITION handle is used to manage partitions of a virtual hard disk in a virtual machine. The handle is obtained by making the PrlVmDevHd_AddPartition (p. 382) or the PrlVmDevHd_GetPartition (p. 385) function call.
PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399)	The PHT_VIRTUAL_DEV_NET_ADAPTER handle is used to manage network adapters in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetNetAdapter (p. 589) function call.
PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425)	The PHT_VIRTUAL_DEV_OPTICAL_DISK handle contains information about a virtual optical disks in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetOpticalDisk (p. 593) function call.
PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425)	The PHT_VIRTUAL_DEV_PARALLEL_PORT handle contains information about a virtual parallel port in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetParallelPort (p. 597) function call.
PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425)	The PHT_VIRTUAL_DEV_SERIAL_PORT handle is used to manage serial ports in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetSerialPort (p. 600) function call.
PHT_VIRTUAL_DEV_SOUND (p. 427)	The PHT_VIRTUAL_DEV_SOUND handle is used to manage sound devices in a virtual machine.
PHT_VIRTUAL_DEV_USB_DEVICE (p. 430)	The PHT_VIRTUAL_DEV_USB handle is used to managing USB devices in a virtual machine.
PHT_VIRTUAL_DISK (p. 432)	The PHT_VIRTUAL_DISK handle is used to create, configure, and manage Virtuozzo virtual disks. A virtual disk is comprised of a set of files including a configuration file containing information about the disk parameters (number of heads, cylinders, sectors, etc.) and the files that are used for data storage. Once created, a virtual disk can be connected to a virtual machine and recognized by it, just like a physical hard disk can be installed in a physical box. To create a new virtual disk use PrlDisk_CreateDisk or PrlDisk_CreateDisk_Local (p. 433). To obtain a handle to an existing virtual disk, use PrlDisk_OpenDisk or... more (p. 432)
PHT_VIRTUAL_DISK_ENCRYPTION (p. 440)	
PHT_VIRTUAL_MACHINE (p. 442)	PHT_VIRTUAL_MACHINE is the main virtual machine management handle. A handle to an existing virtual machine is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetVmList (p. 264) function. To create a new handle, call the PrlSrv_CreateVm (p. 232) function.

PHT_VIRTUAL_NETWORK (p. 507)	The PHT_VIRTUAL_NETWORK handle is used to manage virtual networks. A new handle is obtained by calling the PrlVirtNet_Create (p. 508) function. A handle to an existing virtual network is obtained by calling the PrlSrv_GetVirtualNetworkList (p. 263) function.
PHT_VM_CONFIGURATION (p. 537)	The PHT_VM_CONFIGURATION handle is used to manage the virtual machine configuration. The handle is obtained by calling the PrlVm_GetConfig (p. 464) function.
PHT_VM_GUEST_SESSION (p. 716)	The PHT_VM_GUEST_SESSION handle is used to run programs and execute administrative tasks in the virtual machine. The handle can also be used to obtain some of the guest operating system configuration information. The handle is obtained using the PrlVm_LoginInGuest (p. 473) function.
PHT_VM_INFO (p. 719)	The PHT_VM_INFO handle is a container that contains the virtual machine state, access rights, and other information. The handle is obtained by making the PrlVm_GetState (p. 468) function call.
PHT_VM_TOOLS_INFO (p. 724)	The PHT_VM_TOOLS_INFO handle is used to determine whether the Virtuozzo Tools package is installed in the specified virtual machine and for obtaining its status and version information. The handle is obtained from the job object returned by the PrlVm_GetToolsState (p. 471) function.
PHT_VM_VIRTUAL_DEVICES_INFO (p. 726)	The PHT_VM_VIRTUAL_DEVICES_INFO handle is a container that contains information about the state of the virtual devices in a virtual machine. To obtain the handle, use the PrlVm_GetVirtDevInfo function.

PHT_ACCESS_RIGHTS

The PHT_ACCESS_RIGHTS handle is used to obtain and modify the virtual machine access rights information.

Description

By default, only the owner of a virtual machine has full access to it. All other users are completely unaware of the machine (it will not even appear in the list of the available virtual machines for them). The owner of the virtual machine can change this by setting the virtual machine access rights for other users. The access rights include no-access, view, run, full access (see PRL_VM_ACCESS_FOR_OTHERS (p. 938) for the complete list).

The PHT_ACCESS_RIGHTS handle is obtained by calling the PrlVmCfg_GetAccessRights (p. 552) or the PrlVmInfo_GetAccessRights (p. 720) function. An object returned by either function will contain access rights information for a specific virtual machine. You can use either function to obtain a handle depending on the application logic.

Functions

	Name	Description
	PrlAcl_GetAccessForOthers (p. 10)	Obtains the virtual machine access rights information.
	PrlAcl_GetOwnerName (p. 10)	Determines the virtual machine owner name.
	PrlAcl_IsAllowed (p. 11)	Determines if the current user is authorized to perform a specified task on a virtual machine.
	PrlAcl_IsCurrentSessionOwner (p. 12)	Determines if the current user is the owner of the specified virtual machine.
	PrlAcl_SetAccessForOthers (p. 12)	Sets access rights on a virtual machine.

PrlAcl_GetAccessForOthers Function

Obtains the virtual machine access rights information.

Syntax

```
PRL_RESULT PrlAcl_GetAccessForOthers(
    PRL_HANDLE hAcl,
    PRL_VM_ACCESS_FOR_OTHERS_PTR pnAccessForOthers
);
```

File

PrlApiVm.h

Parameters

hAcl

A handle of type PHT_ACCESS_RIGHTS (p. 9) containing the access rights information for a virtual machine.

pnAccessForOthers

[out] A pointer to a variable that receives the virtual machine access level information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The function determines whether other users besides the owner can view, run, or have full access to a virtual machine.

See Also

PrlAcl_SetAccessForOthers (p. 12)

PrlAcl_GetOwnerName Function

Determines the virtual machine owner name.

Syntax

```
PRL_RESULT PrlAcl_GetOwnerName(
    PRL_HANDLE hAcl,
    PRL_STR sOwnerNameBuf,
    PRL_UINT32_PTR pnOwnerNameBufLength
);
```

File

PrlApiVm.h

Parameters

hAcl

A handle of type PHT_ACCESS_RIGHTS (p. 9) containing the access rights information for a virtual machine.

sOwnerNameBuf

[out] A pointer to a buffer that receives the virtual machine owner name (a UTF-8 encoded, null-terminated string).

pnOwnerNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlAcl_IsCurrentSessionOwner (p. 12)

PrlAcl_IsAllowed Function

Determines if the current user is authorized to perform a specified task on a virtual machine.

Syntax

```
PRL_RESULT PrlAcl_IsAllowed(
    PRL_HANDLE hAcl,
    PRL_ALLOWED_VM_COMMAND nPermission,
    PRL_BOOL_PTR pbAllowed
);
```

File

PrlApiVm.h

Parameters

hAcl

A handle of type PHT_ACCESS_RIGHTS (p. 9) containing the access rights information for a virtual machine.

nPermission

Task type.

pbAllowed

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the user is allowed to perform the task. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlAcl_IsCurrentSessionOwner Function

Determines if the current user is the owner of the specified virtual machine.

Syntax

```
PRL_RESULT PrlAcl_IsCurrentSessionOwner(
    PRL_HANDLE hAcl,
    PRL_BOOL_PTR pbIsOwner
);
```

File

PrlApiVm.h

Parameters

hAcl

A handle of type PHT_ACCESS_RIGHTS (p. 9) containing the access rights information for a virtual machine.

pbIsOwner

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the user is the owner of the virtual machine. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlAcl_SetAccessForOthers Function

Sets access rights on a virtual machine.

Syntax

```
PRL_RESULT PrlAcl_SetAccessForOthers(
    PRL_HANDLE hAcl,
    PRL_VM_ACCESS_FOR_OTHERS nAccessForOthers
);
```

File

PrlApiVm.h

Parameters

hAcl

A handle of type PHT_ACCESS_RIGHTS (p. 9) containing the access rights information for a virtual machine.

nAccessForOthers

The access rights to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The owner of a virtual machine can use this function to grant or deny access rights for the virtual machine to other users. The access rights will be applied to all users. The call will only update the local copy of the PHT_ACCESS_RIGHTS (p. 9) object passed to the function (you will have to properly obtain the object prior to calling this function). To apply the changes to the virtual machine, call the PrlVm_UpdateSecurity (p. 502) function after making this calls and pass the same PHT_ACCESS_RIGHTS (p. 9) handle to it.

See Also

PrlAcl_GetAccessForOthers (p. 10)

PHT_BACKUP

PHT_BACKUP_RESULT

The PHT_BACKUP_RESULT handle is a container that contains the results of a backup operation. The handle is returned by the PrlSrv_CreateVmBackup (p. 233) function call (the handle is returned as the first parameter of the Job object).

Functions

	Name	Description
	PrlBackupResult_GetBackupUuid (p. 13)	Returns the UUID (universally unique ID) of the created backup

PrlBackupResult_GetBackupUuid Function

Returns the UUID (universally unique ID) of the created backup

Syntax

```
PRL_RESULT PrlBackupResult_GetBackupUuid(
    PRL_HANDLE hBackupResult,
    PRL_STR sBackupUuid,
    PRL_UINT32_PTR pnBackupUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hBackupResult

A handle of type PHT_BACKUP_RESULT (p. 13) identifying the information about created backup.

sBackupUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBackupUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_BOOT_DEVICE

The PHT_BOOT_DEVICE handle is used to manage boot devices in a virtual machine. The handle is obtained by calling the PrlVmCfg_GetBootDev (p. 558) or the PrlVmCfg_CreateBootDev (p. 550) function. The first function is used to obtain a handle to an existing device in the boot priority list. The second function creates a new handle and is used to add a new device to the list.

Description

A virtual machine can boot from a variety of devices, such as hard disk drives, floppy drives, removable drives, network etc. The PHT_BOOT_DEVICE object provides functions that can be used to obtain the information about and to modify a device in the boot priority list.

Each device in the boot priority list is uniquely identified by *type* (see PRL_DEVICE_TYPE (p. 1049)) and *index* (a numeric ID). This allows to include more than one device of a particular type in the list (e.g more than one hard disk drive). Please note that each virtual device has an index assigned to it in the main virtual machine configuration. The same index is used to identify a device in the boot priority list.

The position of the device in the list is determined by a *sequence index* assigned to it, starting from 0 and up. The highest boot priority index (the first device to boot from) is 0.

When creating a virtual machine using a default configuration, the boot device priority list is configured automatically.

Note: Any boot device list modification must begin with the PrlVm_BeginEdit (p. 446) and end with the PrlVm_Commit (p. 453) call. These two functions are used to detect collisions with other clients trying to make similar modifications to the same virtual machine. See PrlBootDev_SetSequenceIndex (p. 19) for code sample.

Functions

	Name	Description
	PrlBootDev_GetIndex (p. 15)	Obtains the boot device index.
	PrlBootDev_GetSequenceIndex (p. 15)	Obtains the boot device sequence index.
	PrlBootDev_GetType (p. 16)	Returns the boot device type.
	PrlBootDev_IsInUse (p. 17)	Determines whether the boot device is enabled or disabled.
	PrlBootDev_Remove (p. 17)	Removes the boot device from the boot priority list.
	PrlBootDev_SetIndex (p. 18)	Sets the boot device index.
	PrlBootDev_SetInUse (p. 19)	Enables or disables the boot device.

	PrlBootDev_SetSequenceIndex (p. 19)	Assigns a sequence index to a boot device in the boot priority list.
	PrlBootDev_SetType (p. 21)	Sets the boot device type.

PrlBootDev_GetIndex Function

Obtains the boot device index.

Syntax

```
PRL_RESULT PrlBootDev_GetIndex(
    PRL_HANDLE hVmBootDev,
    PRL_UINT32_PTR pnDevIndex
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

pnDevIndex

[out] A pointer to a variable that receives the device index.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Device index is a property that, together with device type, is used to uniquely identify a device in the boot priority list of a virtual machine. The index should be the same index the device has in the main virtual machine configuration, which is determined using the PrlVmDev_GetIndex (p. 760) function.

See Also

PrlBootDev_SetIndex (p. 18)

PrlBootDev_GetSequenceIndex Function

Obtains the boot device sequence index.

Syntax

```
PRL_RESULT PrlBootDev_GetSequenceIndex(
    PRL_HANDLE hVmBootDev,
    PRL_UINT32_PTR pnSequenceIndex
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

pnSequenceIndex

[out] A pointer to a variable that receives the device sequence index.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The boot device sequence index starts with 0. When a virtual machine is powered on, it will first try to boot from the boot device that has a sequence index of 0. If it cannot boot from the device, it will try to boot from the device with sequence index 1, and so forth.

See Also

PrlBootDev_SetSequenceIndex (p. 19)

PrlBootDev_GetType Function

Returns the boot device type.

Syntax

```
PRL_RESULT PrlBootDev_GetType(
    PRL_HANDLE hVmBootDev,
    PRL_DEVICE_TYPE_PTR pnDevType
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

pnDevType

[out] A pointer to a variable that receives the device type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Device type is a property that, together with device index, is used to uniquely identify a device in the virtual machine boot list. The supported device types are defined in the PRL_DEVICE_TYPE (p. 1049) enumeration.

See Also

PrlBootDev_SetType (p. 21)

PrlBootDev_IsInUse Function

Determines whether the boot device is enabled or disabled.

Syntax

```
PRL_RESULT PrlBootDev_IsInUse(
    PRL_HANDLE hVmBootDev,
    PRL_BOOL_PTR pbInUse
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

pbInUse

[out] Pointer to a variable that receives the result PRL_TRUE (p. 1652) indicates that the device is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A boot device can be either enabled or disabled. If a device is disabled, it is ignored during the boot operation. To change the status of a device, use the PrlBootDev_SetInUse (p. 19) function.

See Also

PrlBootDev_SetInUse (p. 19)

PrlBootDev_Remove Function

Removes the boot device from the boot priority list.

Syntax

```
PRL_RESULT PrlBootDev_Remove(
    PRL_HANDLE hVmBootDev
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or the device has been removed from the boot list.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

This function permanently removes a devices from the list. If you would like to temporarily disable a device, use the PrlBootDev_SetInUse (p. 19) function. To add a device back to the list, use PrlVmCfg_CreateBootDev (p. 550) and then call the necessary functions of the PHT_BOOT_DEVICE (p. 14) handle to set the necessary properties.

PrlBootDev_SetIndex Function

Sets the boot device index.

Syntax

```
PRL_RESULT PrlBootDev_SetIndex(
    PRL_HANDLE hVmBootDev,
    PRL_UINT32 nDevIndex
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

nDevIndex

The index of the device of interest as it exists in the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or the device has been removed from the boot list.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Device index is a property that, together with device type, is used to uniquely identify a device in the virtual machine boot priority list. The index must be the same index the device has in the main virtual machine configuration or it will not be recognized during the boot procedure. To obtain a virtual device index from the virtual machine configuration, use the PrlVmDev_GetIndex (p. 760) function.

See Also

PrlBootDev_GetIndex (p. 15)

PrlBootDev_SetInUse Function

Enables or disables the boot device.

Syntax

```
PRL_RESULT PrlBootDev_SetInUse(
    PRL_HANDLE hVmBootDev,
    PRL_BOOL bInUse
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

bInUse

A boolean value specifying the action type. PRL_TRUE (p. 1652) enables a device, PRL_FALSE (p. 1626) disables it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or the device has been removed from the boot list.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A boot device can be enabled or disabled without being removed from the boot device priority list. If the device is disabled, it is ignored during the boot operation. To obtain the current status of a device, use the PrlBootDev_IsInUse (p. 17) function. To permanently remove a device from the list, use the PrlBootDev_Remove (p. 17) function.

See Also

PrlBootDev_IsInUse (p. 17)

PrlBootDev_Remove (p. 17)

PrlBootDev_SetSequenceIndex Function

Assigns a sequence index to a boot device in the boot priority list.

Syntax

```
PRL_RESULT PrlBootDev_SetSequenceIndex(
    PRL_HANDLE hVmBootDev,
    PRL_UINT32 nSequenceIndex
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

nSequenceIndex

The boot sequence index to set for the device. The value of 0 (zero) is the highest boot priority and indicates the first device to boot from. The value of 1 (one) indicates the second device to boot from, and so forth.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or the device has been removed from the boot list.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The boot device sequence index starts with 0. When a virtual machine is powered on, it will first try to boot from the boot device that has a sequence index of 0. If it cannot boot from the device, it will try to boot from the device with sequence index 1, and so forth.

If you are changing a sequence index for an existing device, this is the only function that you have to call. If you are adding a new boot device to the list, you will have to set all the other properties, including the device index, type, and enabled/disabled flag. Please note that each device in the boot priority list must have a unique sequence index, so if you are modifying a sequence index for an existing device or assigning an index to a new device, you will have to make sure that no two devices have the same index.

See Also

PrlBootDev_GetSequenceIndex (p. 15)

Example

```
PRL_HANDLE hJobBeginEdit;
PRL_HANDLE hJobCommit;
PRL_RESULT nJobRetCode;

// Timestamp the beginning of the configuration changes operation.
hJobBeginEdit = PrlVm_BeginEdit(hVm);
ret = PrlJob_Wait(hJobBeginEdit, 10000);
PrlJob_GetRetCode(hJobBeginEdit, &nJobRetCode);
if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJobBeginEdit);
    return nJobRetCode;
}

// Modify boot options.
// Set boot device list as follows:
// 0. CD/DVD drive.
// 1. Hard disk.
// 2. Network adapter.
// 3. Floppy disk drive.
//
PRL_UINT32 nDevCount; // Device count.
PRL_HANDLE hDevice;   // A handle to the device.
PRL_DEVICE_TYPE devType; // Device type.

// Get the total number of devices.
ret = PrlVm_GetBootDevCount(hVm, &nDevCount);

// Iterate through the device list.
```

```
// Get a handle for each available device.
// Set an index for a device in the boot list.
for (int i = 0; i < nDevCount; ++i)
{
    ret = PrlVm_GetBootDev(hVm, i, &hDevice);
    ret = PrlBootDev_GetType(hDevice, &devType);

    if (devType == PDE_OPTICAL_DISK)
    {
        PrlBootDev_SetSequenceIndex(hDevice, 0);
    }
    if (devType == PDE_HARD_DISK)
    {
        PrlBootDev_SetSequenceIndex(hDevice, 1);
    }
    else if (devType == PDE_GENERIC_NETWORK_ADAPTER)
    {
        PrlBootDev_SetSequenceIndex(hDevice, 2);
    }
    else if (devType == PDE_FLOPPY_DISK)
    {
        PrlBootDev_SetSequenceIndex(hDevice, 3);
    }
    else
    {
        PrlBootDev_Remove(hDevice);
    }
}

// Commit the changes.
hJobCommit = PrlVm_Commit(hVm);

// Check the results of the commit operation.
ret = PrlJob_Wait(hJobCommit, 10000);
PrlJob_GetRetCode(hJobCommit, &nJobRetCode);
if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Commit error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJobCommit);
    return nJobRetCode;
}
```

PrlBootDev_SetType Function

Sets the boot device type.

Syntax

```
PRL_RESULT PrlBootDev_SetType(
    PRL_HANDLE hVmBootDev,
    PRL_DEVICE_TYPE nDevType
);
```

File

PrlApiVm.h

Parameters

hVmBootDev

A handle of type PHT_BOOT_DEVICE (p. 14) identifying the boot device.

nDevType

The device type to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or the device has been removed from the boot list.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when adding a device to the boot device priority list. Device type is a property that, together with device index, is used to uniquely identify a device in a virtual machine boot list. When adding a device to the boot priority list, these properties must be set correctly or the device will not be recognized. The supported device types are defined in the PRL_DEVICE_TYPE (p. 1049) enumeration.

See Also

PrlBootDev_GetType (p. 16)

PHT_CPU_FEATURES

Functions

	Name	Description
	PrlCpuFeatures_Create (p. 22)	Creates a new handle of type PHT_CPU_FEATURES.
	PrlCpuFeatures_GetValue (p. 23)	Returns value of the specific cpu feature.
	PrlCpuFeatures_SetValue (p. 23)	Sets value of the specific feature.

PrlCpuFeatures_Create Function

Creates a new handle of type PHT_CPU_FEATURES (p. 22).

Syntax

```
PRL_RESULT PrlCpuFeatures_Create(
    PRL_HANDLE_PTR phCpuFeatures
);
```

File

PrlApiDisp.h

Parameters

phCpuFeatures

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCpuFeatures_GetValue Function

Returns value of the specific cpu feature.

Syntax

```
PRL_RESULT PrlCpuFeatures_GetValue(
    PRL_HANDLE hCpuFeatures,
    PRL_CPU_FEATURES_EX nId,
    PRL_UINT32_PTR pnValue
);
```

File

PrlApiDisp.h

Parameters

hCpuFeatures

A handle of type PHT_CPU_FEATURES (p. 22).

nId

Id of specific feature.

pnValue

Pointer for storing result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer or invalid feature id was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlCpuFeatures_SetValue (p. 23)

PrlCpuFeatures_SetValue Function

Sets value of the specific feature.

Syntax

```
PRL_RESULT PrlCpuFeatures_SetValue(
    PRL_HANDLE hCpuFeatures,
    PRL_CPU_FEATURES_EX nId,
    PRL_UINT32 nValue
);
```

File

PrlApiDisp.h

Parameters

hCpuFeatures

A handle of type PHT_CPU_FEATURES (p. 22).

nId

Id of specific feature.

nValue

Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid feature id was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlCpuFeatures_GetValue (p. 23)

PHT_CPU_POOL

Functions

	Name	Description
	PrlCPUPool_GetCpuFeaturesMask (p. 24)	Obtains a handle to cpu pool information
	PrlCPUPool_GetName (p. 25)	Returns the vendor of the pool
	PrlCPUPool_GetVendor (p. 25)	Returns the name of the pool

PrlCPUPool_GetCpuFeaturesMask Function

Obtains a handle to cpu pool information

Syntax

```
PRL_RESULT PrlCPUPool_GetCpuFeaturesMask(
    PRL_HANDLE hCpuPool,
    PRL_HANDLE_PTR phCpuFeatures
);
```

File

PrlApiDisp.h

Parameters

hCpuPool

A handle of type PHT_CPU_POOL (p. 24)

ph

[out] A pointer to a variable that receives the handle. The type of the handle is PHT_CPU_FEATURES (p. 22)

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate new object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCPUPool_GetName Function

Returns the vendor of the pool

Syntax

```
PRL_RESULT PrlCPUPool_GetName(
    PRL_HANDLE hCpuPool,
    PRL_STR sBuf,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hCpuPool

A handle of type PHT_CPU_POOL (p. 24)

sBuf

[out] Pointer to a buffer that receives the pool name (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCPUPool_GetVendor Function

Returns the name of the pool

Syntax

```
PRL_RESULT PrlCPUPool_GetVendor(
    PRL_HANDLE hCpuPool,
    PRL_STR sBuf,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hCpuPool

A handle of type PHT_CPU_POOL (p. 24)

sBuf

[out] Pointer to a buffer that receives the pool vendor (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_CT_TEMPLATE

The PHT_CT_TEMPLATE handle is used to manage operating system templates on which Containers are based.

Functions

	Name	Description
	PrlCtTemplate_GetCpuMode (p. 26)	Determines the specified template CPU mode (32 bit or 64 bit).
	PrlCtTemplate_GetDescription (p. 27)	Returns description of the Container's template.
	PrlCtTemplate_GetName (p. 28)	Returns name of the Container's template.
	PrlCtTemplate_GetOsTemplate (p. 28)	Returns the version of the operating system template for which the specified application template is.
	PrlCtTemplate_GetOsType (p. 29)	Returns the type of the operating system for which the specified template is.
	PrlCtTemplate_GetOsVersion (p. 30)	Returns the version of the operating system for which the specified template is running.
	PrlCtTemplate_GetType (p. 30)	Returns the type of the specified template.
	PrlCtTemplate_GetVersion (p. 31)	Returns version of the Container's template.
	PrlCtTemplate_IsCached (p. 31)	Determines whether the template cached.

PrlCtTemplate_GetCpuMode Function

Determines the specified template CPU mode (32 bit or 64 bit).

Syntax

```
PRL_RESULT PrlCtTemplate_GetCpuMode(
    PRL_HANDLE hTemplate,
    PRL_CPU_MODE_PTR pnCpuMode
```

```
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

pnCpuMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the template object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCtTemplate_GetDescription Function

Returns description of the Container's template.

Syntax

```
PRL_RESULT PrlCtTemplate_GetDescription(
    PRL_HANDLE hTemplate,
    PRL_STR sDescription,
    PRL_UINT32_PTR pnDescriptionLength
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

sDescription

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDescriptionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCtTemplate_GetName Function

Returns name of the Container's template.

Syntax

```
PRL_RESULT PrlCtTemplate_GetName(
    PRL_HANDLE hTemplate,
    PRL_STR sName,
    PRL_UINT32_PTR pnNameLength
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

sName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCtTemplate_GetOsTemplate Function

Returns the version of the operating system template for which the specified application template is.

Syntax

```
PRL_RESULT PrlCtTemplate_GetOsTemplate(
    PRL_HANDLE hTemplate,
    PRL_STR sOsTemplate,
    PRL_UINT32_PTR pnOsTemplateLength
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

sOsTemplate

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnOsTemplateLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCtTemplate_GetOsType Function

Returns the type of the operating system for which the specified template is.

Syntax

```
PRL_RESULT PrlCtTemplate_GetOsType(
    PRL_HANDLE hTemplate,
    PRL_UINT32_PTR pnOsType
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

pnOsType

[out] A pointer to a variable that receives the OS type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the OS information is not present in the template configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The OS types are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_TYPE_ (e.g. PVS_GUEST_TYPE_LINUX (p. 1582)).

PrlCtTemplate_GetOsVersion Function

Returns the version of the operating system for which the specified template is running.

Syntax

```
PRL_RESULT PrlCtTemplate_GetOsVersion(
    PRL_HANDLE hTemplate,
    PRL_UINT32_PTR pnOsVersion
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

pnOsVersion

[out] A pointer to a variable that receives the OS version.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the OS information is not present in the template configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The OS versions are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_LIN_REDHAT (p. 1594)).

PrlCtTemplate_GetType Function

Returns the type of the specified template.

Syntax

```
PRL_RESULT PrlCtTemplate_GetType(
    PRL_HANDLE hTemplate,
    PRL_CT_TEMPLATE_TYPE_PTR pType
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

pnType

[out] A pointer to a variable that receives the type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCtTemplate_GetVersion Function

Returns version of the Container's template.

Syntax

```
PRL_RESULT PrlCtTemplate_GetVersion(
    PRL_HANDLE hTemplate,
    PRL_STR sVersion,
    PRL_UINT32_PTR pnVersionLength
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

sVersion

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVersionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlCtTemplate_IsCached Function

Determines whether the template cached.

Syntax

```
PRL_RESULT PrlCtTemplate_IsCached(
    PRL_HANDLE hTemplate,
    PRL_BOOL_PTR pbCached
);
```

File

PrlApiVm.h

Parameters

hTemplate

A handle of type PHT_CT_TEMPLATE (p. 26).

pbCached

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the template object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Makes sense only for operating system templates.

PHT_CVSRC

The PHT_CVSRC handle is used to capture data from a video source on the host computer and display in the guest operating system.

PHT_DESKTOP_CONTROL

The PHT_DESKTOP_CONTROL handle is used to remotely access the desktop of the physical host computer. Use the PrlSrv_CreateDesktopControl to create the handle. See the **Remote Desktop Access** section for the list of functions that support this functionality.

PHT_DISP_CONFIG

The PHT_DISP_CONFIG handle is used to manage Virtuozzo Service preferences. The handle is obtained by calling the PrlSrv_GetCommonPrefs (p. 244) function.

Description

Virtuozzo Service preferences include memory, network, default virtual machine folder, and some other settings. The following describes how to modify the memory and network settings as a demonstration of the handle usage.

First, you have to obtain a handle of type PHT_DISP_CONFIG by making the PrlSrv_GetCommonPrefs (p. 244) function call.

To make the memory modification:

1. Call PrlSrv_CommonPrefsBeginEdit (p. 226) to mark the beginning of the operation.
2. Call one or more of the memory management functions from the PHT_DISP_CONFIG group to modify the desired settings.
3. Call PrlSrv_CommonPrefsCommit (p. 227) to save the changes to the Virtuozzo Service configuration.

The network adapter changes are performed outside of the begin/commit block. To modify an existing or to create a new virtual network adapter, use the following steps:

1. First, obtain a handle of type PHT_DISP_NET_ADAPTER by calling the PrlDispCfg_GetDispNet or the PrlDispCfg_CreateDispNet function. The first function obtains a handle to an existing adapter and is used when modifying an adapter. The second function creates and returns a new handle and is used when adding a new virtual adapter to the Virtuozzo Service configuration.
2. Populate or modify the necessary network adapter properties using functions from the PHT_DISP_NET_ADAPTER group.
3. If you are creating a new adapter, call PrlSrv_AddNetAdapter to add the adapter to the Virtuozzo Service configuration. If you are modifying an existing adapter, call PrlSrv_UpdateNetAdapter to save the changes. If you would like to delete an adapter from the Virtuozzo Service configuration, call the PrlSrv_DeleteNetAdapter function.

Functions

	Name	Description
	PrlDispCfg_ArePluginsEnabled (p. 35)	Determines whether the plugins enabled or not
	PrlDispCfg_CanChangeDefaultSettings (p. 36)	Determines if new users have the right to modify Virtuozzo Service preferences.
	PrlDispCfg_EnablePlugins (p. 36)	Enable or disable the plugins support
	PrlDispCfg_GetBackupTimeout (p. 37)	Get backup timeout in seconds.
	PrlDispCfg_GetBackupTmpDir (p. 37)	Returns the backup temporary directory.
	PrlDispCfg_GetBackupUserLogin (p. 38)	Returns the backup user login.
	PrlDispCfg_GetConfirmationsList (p. 39)	Obtains a handles list of operations which required administrator confirmation (for all users and for all virtual machines(if operation is applicable to vm))
	PrlDispCfg_GetCpuFeaturesMaskEx (p. 39)	Allows to get the CPU features masks from dispatcher config.
	PrlDispCfg_GetCpuPool (p. 40)	Obtains a handle to cpu pool information
	PrlDispCfg_GetDefaultBackupDirectory (p. 40)	Returns PATH to the default backup directory.
	PrlDispCfg_GetDefaultBackupServer (p. 41)	Returns the default backup server host name or IP address.
	PrlDispCfg_GetDefaultCtDir (p. 42)	Obtains path of the directory in which new Container are created by default.
	PrlDispCfg_GetDefaultVmDir (p. 42)	Obtains name and path of the directory in which new virtual machines are created by default.
	PrlDispCfg_GetDefaultVNCHostName (p. 43)	Returns the default VNC host name for the Virtuozzo Service.
	PrlDispCfg_GetMaxReservMemLimit (p. 44)	Determines the maximum amount of physical memory that can be reserved for Virtuozzo Service operation.
	PrlDispCfg_GetMaxVmMem (p. 45)	Determines the maximum memory size that can be allocated to an individual virtual machine.
	PrlDispCfg_GetMinReservMemLimit (p. 45)	Determines the minimum amount of physical memory that must be reserved for Virtuozzo Service operation.
	PrlDispCfg_GetMinSecurityLevel (p. 46)	Determines the lowest allowable security level that can be used to connect to the Virtuozzo Service.

	PrlDispCfg_GetMinVmMem (p. 47)	Determines the minimum required memory size that must be allocated to an individual virtual machine.
	PrlDispCfg_GetRecommendMaxVmMem (p. 48)	Determines the recommended memory size for an individual virtual machine.
	PrlDispCfg_GetReservedMemLimit (p. 49)	Determines the amount of physical memory reserved for Virtuozzo Service operation.
	PrlDispCfg_GetUsbIdentity (p. 49)	Returns handle of type PHT_USB_IDENTITY (p. 368) storing information of USB device with specified index.
	PrlDispCfg_GetUsbIdentityCount (p. 50)	Determines number of USB devices known to the Virtuozzo Service.
	PrlDispCfg_GetVmCpuLimitType (p. 51)	%VM_ONLY% Returns VM CPU limit type.
	PrlDispCfg_GetVNCBasePort (p. 51)	Obtains the currently set base VNC port number.
	PrlDispCfg_IsAdjustMemAuto (p. 52)	Determines whether memory allocation for Virtuozzo Service is performed automatically or manually.
	PrlDispCfg_IsAllowMultiplePMC (p. 53)	Determines whether the Allow Multiple PMC option is enabled.
	PrlDispCfg_IsBackupUserPasswordEnabled (p. 53)	Returns the backup user password enabling sign.
	PrlDispCfg_IsLogRotationEnabled (p. 54)	Determines whether the log rotation is enabled for dispatcher and virtual machine logs.
	PrlDispCfg_IsSendStatisticReport (p. 54)	Determines whether send statistics reports (CEP) mechanism activated.
	PrlDispCfg_IsVerboseLogEnabled (p. 55)	Determines whether the verbose log level is configured for dispatcher and virtual machines processes.
	PrlDispCfg_SetAdjustMemAuto (p. 55)	Allows to choose the Virtuozzo Service memory allocation mode from automatic or manual.
	PrlDispCfg_SetAllowMultiplePMC (p. 56)	Enables or disables the Allow Multiple PMC option.
	PrlDispCfg_SetBackupTimeout (p. 57)	Set backup timeout in seconds.
	PrlDispCfg_SetBackupTmpDir (p. 57)	Sets the backup temporary directory.
	PrlDispCfg_SetBackupUserLogin (p. 58)	Sets the backup user login.
	PrlDispCfg_SetBackupUserPassword (p. 58)	Sets the backup user password.
	PrlDispCfg_SetBackupUserPasswordEnabled (p. 59)	Sets the backup user password.
	PrlDispCfg_SetCanChangeDefaultSettings (p. 59)	Allows to grant or deny a permission to new users to modify Virtuozzo Service preferences.
	PrlDispCfg_SetConfirmationsList (p. 60)	Obtains a handles list of operations which required administrator confirmation (for all users and for all virtual machines(if operation is applicable to vm))
	PrlDispCfg_SetCpuFeaturesMaskEx (p. 61)	Allows to set the CPU features masks.
	PrlDispCfg_SetDefaultBackupDirectory (p. 61)	Sets PATH to the default backup directory.
	PrlDispCfg_SetDefaultBackupServer (p. 62)	Sets the default backup server host name or IP address.
	PrlDispCfg_SetDefaultVNCHostName (p. 62)	Sets the base VNC host name.
	PrlDispCfg_SetLogRotationEnabled (p. 63)	Enables or disables the log rotation for dispatcher and virtual machine logs.
	PrlDispCfg_SetMaxReservMemLimit (p. 64)	Allows to manually set the upper limit of the memory size that can be reserved for Virtuozzo Service operation.
	PrlDispCfg_SetMaxVmMem (p. 64)	Allows to set the maximum memory size that can be allocated to an individual virtual machine.

	PrlDispCfg_SetMinReservMemLimit (p. 65)	Allows to manually set the lower limit of the memory size that must be reserved for Virtuozzo Service operation.
	PrlDispCfg_SetMinSecurityLevel (p. 66)	Allows to set the lowest allowable security level that can be used to connect to the specified Virtuozzo Service.
	PrlDispCfg_SetMinVmMem (p. 67)	Allows to set the minimum required memory size that must be allocated to an individual virtual machine.
	PrlDispCfg_SetRecommendMaxVmMem (p. 68)	Allows to manually set recommended memory size for an individual virtual machine.
	PrlDispCfg_SetReservedMemLimit (p. 68)	Allows to set the amount of memory that will be allocated for Virtuozzo Service operation.
	PrlDispCfg_SetSendStatisticReport (p. 69)	Allows to switch on/off mechanism of sending statistics reports (CEP).
	PrlDispCfg_SetUsbIdentAssociation (p. 70)	(Un)Associates USB device with given Vm.
	PrlDispCfg_SetVerboseLogEnabled (p. 71)	Enables or disables the verbose log level for dispatcher and virtual machines processes.
	PrlDispCfg_SetVmCpuLimitType (p. 71)	%VM_ONLY%
	PrlDispCfg_SetVNCBasePort (p. 72)	Sets the base VNC port number.

PrlDispCfg_ArePluginsEnabled Function

Determines whether the plugins enabled or not

Syntax

```
PRL_RESULT PrlDispCfg_ArePluginsEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbIsPluginsAllowed
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbIsPluginsAllowed

[out] Pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the plugins support is allowed PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_CanChangeDefaultSettings Function

Determines if new users have the right to modify Virtuozzo Service preferences.

Syntax

```
PRL_RESULT PrlDispCfg_CanChangeDefaultSettings(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbDefaultChangeSettings
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbDefaultChangeSettings

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that new users are allowed to change the Virtuozzo Service preferences. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, only administrators of the host machine can modify Virtuozzo Service preferences. When a new Virtuozzo Service user profile is created (this happens automatically when a user logs in to the Virtuozzo Service for the first time), he/she will be granted or denied this privilege based on the flag set in the Virtuozzo Service configuration. This function determines the status of this flag.

See Also

PrlDispCfg_SetCanChangeDefaultSettings (p. 59)

PrlDispCfg_EnablePlugins Function

Enable or disable the plugins support

Syntax

```
PRL_RESULT PrlDispCfg_EnablePlugins(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bEnablePluginsSupport
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bEnablePluginsSupport

[in] A value to set. Use PRL_TRUE (p. 1652) to enable, PRL_FALSE (p. 1626) to disable plugins support.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetBackupTimeout Function

Get backup timeout in seconds.

Syntax

```
PRL_RESULT PrlDispCfg_GetBackupTimeout(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnBackupTimeout
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnTimeout

[out] A pointer to a buffer that receives the result

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetBackupTmpDir Function

Returns the backup temporary directory.

Syntax

```
PRL_RESULT PrlDispCfg_GetBackupTmpDir(
    PRL_HANDLE hDispConfig,
    PRL_STR sBackupTmpDir,
    PRL_UINT32_PTR pnBackupTmpDirBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sBackupTmpDir

[out] Pointer to a buffer that receives the backup temporary directory (a UTF-8 encoded, null-terminated string).

pnBackupTmpDirBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetBackupUserLogin Function

Returns the backup user login.

Syntax

```
PRL_RESULT PrlDispCfg_GetBackupUserLogin(
    PRL_HANDLE hDispConfig,
    PRL_STR sUserLogin,
    PRL_UINT32_PTR pnUserLoginBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sUserLogin

[out] Pointer to a buffer that receives the backup user login (a UTF-8 encoded, null-terminated string).

pnUserLoginBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetConfirmationsList Function

Obtains a handles list of operations which required administrator confirmation (for all users and for all virtual machines(if operation is applicable to vm))

Syntax

```
PRL_RESULT PrlDispCfg_GetConfirmationsList(
    PRL_HANDLE hDispConfig,
    PRL_HANDLE_PTR phConfirmList
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

phConfirmList

[out] A pointer to a variable that receives the handle. The type of the handle is PHT_OPAQUE_TYPE_LIST (p. 177) (values in list consist in enum PRL_ALLOWED_VM_COMMAND (p. 1036))

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a handle.

See Also

PrlSrv_DisableConfirmationMode (p. 236)() PrlSrv_EnableConfirmationMode (p. 236)() PrlSrv_IsConfirmationModeEnabled (p. 268)() PrlVmCfg_GetConfirmationsList (p. 561)() PrlVmCfg_SetConfirmationsList (p. 660)() PrlDispCfg_GetConfirmationsList() PrlDispCfg_SetConfirmationsList (p. 60)()

PrlDispCfg_GetCpuFeaturesMaskEx Function

Allows to get the CPU features masks from dispatcher config.

Syntax

```
PRL_RESULT PrlDispCfg_GetCpuFeaturesMaskEx(
    PRL_HANDLE hDispConfig,
    PRL_HANDLE_PTR phCpuFeatures
);
```

File

PrlApiDisp.h

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate new object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

hDispConfig : A handle of type PHT_DISP_CONFIG (p. 32). phCpuFeatures : A pointer to a variable that receives a handle of type PHT_CPU_FEATURES (p. 22).

See Also

PrlDispCfg_SetCpuFeaturesMaskEx (p. 61)

PrlDispCfg_GetCpuPool Function

Obtains a handle to cpu pool information

Syntax

```
PRL_RESULT PrlDispCfg_GetCpuPool(
    PRL_HANDLE hDispConfig,
    PRL_HANDLE_PTR phCpuPool
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

phCpuPool

[out] A pointer to a variable that receives the handle. The type of the handle is PHT_CPU_POOL (p. 24). If node is not in cpu pool handle will be PRL_INVALID_HANDLE (p. 1627)

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetDefaultBackupDirectory Function

Returns PATH to the default backup directory.

Syntax

```
PRL_RESULT PrlDispCfg_GetDefaultBackupDirectory(
    PRL_HANDLE hDispConfig,
    PRL_STR sBackupDirectory,
    PRL_UINT32_PTR pnBackupDirectoryBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sBackupDirectory

[out] Pointer to a buffer that receives PATH to the backup directory (a UTF-8 encoded, null-terminated string).

pnBackupDirectoryBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetDefaultBackupServer Function

Returns the default backup server host name or IP address.

Syntax

```
PRL_RESULT PrlDispCfg_GetDefaultBackupServer(
    PRL_HANDLE hDispConfig,
    PRL_STR sBackupServer,
    PRL_UINT32_PTR pnBackupServerBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sBackupServer

[out] Pointer to a buffer that receives the backup server host name or IP address (a UTF-8 encoded, null-terminated string).

pnBackupServerBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetDefaultCtDir Function

Obtains path of the directory in which new Container are created by default.

Syntax

```
PRL_RESULT PrlDispCfg_GetDefaultCtDir(
    PRL_HANDLE hDispConfig,
    PRL_STR sDefaultDir,
    PRL_UINT32_PTR pnDefaultDirBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sDefaultDir

[out] A pointer to a buffer that receives the directory name (a UTF-8 encoded, null-terminated string).

pnDefaultDirBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetDefaultVmDir Function

Obtains name and path of the directory in which new virtual machines are created by default.

Syntax

```
PRL_RESULT PrlDispCfg_GetDefaultVmDir(
    PRL_HANDLE hDispConfig,
    PRL_STR sDefaultVmDir,
    PRL_UINT32_PTR pnDefaultVmDirBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sDefaultVmDir

[out] A pointer to a buffer that receives the directory name (a UTF-8 encoded, null-terminated string).

pnDefaultVmDirBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The virtual machine directory returned by this function is the default directory for all users and cannot be modified. Individual users can have their own default virtual machine directories if needed. To set a default directory for a user, use the PrlUsrCfg_SetDefaultVmFolder (p. 378) function. If a user has its own virtual machine directory set up, it takes precedence over the directory returned by this function.

PrlDispCfg_GetDefaultVNCHostName Function

Returns the default VNC host name for the Virtuozzo Service.

Syntax

```
PRL_RESULT PrlDispCfg_GetDefaultVNCHostName(
    PRL_HANDLE hDispConfig,
    PRL_STR sHostName,
    PRL_UINT32_PTR pnHostNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sHostName

[out] Pointer to a buffer that receives the VNC host name (a UTF-8 encoded, null-terminated string).

pnHostNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetMaxReservMemLimit Function

Determines the maximum amount of physical memory that can be reserved for Virtuozzo Service operation.

Syntax

```
PRL_RESULT PrlDispCfg_GetMaxReservMemLimit(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMemSize

[out] A pointer to a variable that receives the memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, memory limits and the actual size of the reserved memory are calculated automatically based on the total RAM available on the host machine. If needed, you can modify the default values. When setting the size of the memory reserved for Virtuozzo Service, use this function to find out the maximum allowable size.

See Also

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_GetMaxVmMem Function

Determines the maximum memory size that can be allocated to an individual virtual machine.

Syntax

```
PRL_RESULT PrlDispCfg_GetMaxVmMem(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMemSize

[out] A pointer to a variable that receives the memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The actual amount of memory for a particular virtual machine is set in the virtual machine configuration and cannot exceed the value returned by this function.

See Also

PrlDispCfg_GetMinVmMem (p. 47)

PrlDispCfg_GetRecommendMaxVmMem (p. 48)

PrlDispCfg_SetMaxVmMem (p. 64)

PrlDispCfg_SetMinVmMem (p. 67)

PrlDispCfg_SetRecommendMaxVmMem (p. 68)

PrlDispCfg_GetMinReservMemLimit Function

Determines the minimum amount of physical memory that must be reserved for Virtuozzo Service operation.

Syntax

```
PRL_RESULT PrlDispCfg_GetMinReservMemLimit(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMemSize

[out] A pointer to a variable that receives the memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, memory limits and the actual size of the reserved memory are calculated automatically based on the total RAM available on the host machine. If needed, you can modify the default values. When setting the size of the memory reserved on the host machine for Virtuozzo Service, use this function to find out the minimum required size.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_GetMinSecurityLevel Function

Determines the lowest allowable security level that can be used to connect to the Virtuozzo Service.

Syntax

```
PRL_RESULT PrlDispCfg_GetMinSecurityLevel(
    PRL_HANDLE hDispConfig,
    PRL_SECURITY_LEVEL_PTR pnMinSecurityLevel
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMinSecurityLevel

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Service has a configuration parameter that specifies the connection security level that must be used by clients to communicate with it. If a client tries to login using a lower level than the one returned by this function, the connection to the Virtuozzo Service will be refused. Higher security levels are allowed. If the minimum security level is not known in advance, use this function to determine it before establishing a connection with the Service.

See Also

PrlDispCfg_SetMinSecurityLevel (p. 66)

PrlSrv_Login (p. 272)

PrlSrv_LoginLocal (p. 276)

PrlDispCfg_GetMinVmMem Function

Determines the minimum required memory size that must be allocated to an individual virtual machine.

Syntax

```
PRL_RESULT PrlDispCfg_GetMinVmMem(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMemSize

[out] A pointer to a variable that receives the memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The value returned by this function indicates the minimum amount of memory that is absolutely required for a virtual machine to run. The actual amount of memory for a particular virtual machine is set in the virtual machine configuration and cannot be less than the value returned by this function.

See Also

PrlDispCfg_GetMaxVmMem (p. 45)

PrlDispCfg_GetRecommendMaxVmMem (p. 48)

PrlDispCfg_SetMaxVmMem (p. 64)

PrlDispCfg_SetMinVmMem (p. 67)

PrlDispCfg_SetRecommendMaxVmMem (p. 68)

PrlDispCfg_GetRecommendMaxVmMem Function

Determines the recommended memory size for an individual virtual machine.

Syntax

```
PRL_RESULT PrlDispCfg_GetRecommendMaxVmMem(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMemSize

[out] A pointer to a variable that receives the memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Setting the virtual machine memory size to the value returned by this function is the safest way to do it but is not mandatory. You can use your own value but you have to make sure that it falls within the limits returned by the PrlDispCfg_GetMaxVmMem (p. 45) and PrlDispCfg_GetMinVmMem (p. 47) functions.

See Also

PrlDispCfg_GetMaxVmMem (p. 45)

PrlDispCfg_GetMinVmMem (p. 47)

PrlDispCfg_SetMaxVmMem (p. 64)

PrlDispCfg_SetMinVmMem (p. 67)

PrlDispCfg_SetRecommendMaxVmMem (p. 68)

PrlDispCfg_GetReservedMemLimit Function

Determines the amount of physical memory reserved for Virtuozzo Service operation.

Syntax

```
PRL_RESULT PrlDispCfg_GetReservedMemLimit(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnMemSize

[out] A pointer to a variable that receives the memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, this memory size is calculated automatically based on the RAM size available on the host machine. If needed, the default value can be modified manually using the PrlDispCfg_SetReservedMemLimit (p. 68) function.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_GetUsbIdentity Function

Returns handle of type PHT_USB_IDENTITY (p. 368) storing information of USB device with specified index.

Syntax

```
PRL_RESULT PrlDispCfg_GetUsbIdentity(
    PRL_HANDLE hDispCfg,
```

```
    PRL_UINT32 nUsbIdentIndex,
    PRL_HANDLE_PTR phUsbIdent
);
```

File

PrlApiDisp.h

Parameters

hDispCfg

A handle of type PHT_DISP_CONFIG (p. 32).

nUsbIdentIndex

A numeric index identifying object with information about USB device. To determine total number of such objects use PrlDispCfg_GetUsbIdentityCount (p. 50)

phUsbIdent

[out] a pointer to a buffer receiving PHT_USB_IDENTITY (p. 368) handle

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle, wrong index or null pointer was passed

PRL_ERR_OUT_OF_MEMORY (p. 1334) - insufficient memory to complete operation

PRL_ERR_SUCCESS (p. 1365) - function completed successively

PrlDispCfg_GetUsbIdentityCount Function

Determines number of USB devices known to the Virtuozzo Service.

Syntax

```
PRL_RESULT PrlDispCfg_GetUsbIdentityCount(
    PRL_HANDLE hDispCfg,
    PRL_UINT32_PTR pnUsbIdentityCount
);
```

File

PrlApiDisp.h

Parameters

hDispCfg

A handle of type PHT_DISP_CONFIG (p. 32).

pnUsbIdentityCount

[out] A pointer to a variable that receives result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - either hDispCfg is not a valid PHT_DISP_CONFIG (p. 32)

PRL_SUCCESS - function completed successfully

PrlDispCfg_GetVmCpuLimitType Function

%VM_ONLY%

Returns VM CPU limit type.

Syntax

```
PRL_RESULT PrlDispCfg_GetVmCpuLimitType(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnVmCpuLimitType
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnVmCpuLimitType

[out] Pointer to a buffer that receives Vm CPU limit type

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_GetVNCBasePort Function

Obtains the currently set base VNC port number.

Syntax

```
PRL_RESULT PrlDispCfg_GetVNCBasePort(
    PRL_HANDLE hDispConfig,
    PRL_UINT32_PTR pnPort
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pnPort

[out] A pointer to a variable that receives the port number.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_IsAdjustMemAuto Function

Determines whether memory allocation for Virtuozzo Service is performed automatically or manually.

Syntax

```
PRL_RESULT PrlDispCfg_IsAdjustMemAuto(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbAdjustMemAuto
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbAdjustMemAuto

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the memory is set automatically. PRL_FALSE (p. 1626) indicates that the size is set manually.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, the amount of memory allocated for Virtuozzo Service is calculated automatically based on the total RAM available on the host. You can allocate a different amount of memory if needed using the PrlDispCfg_SetReservedMemLimit (p. 68) function. Before you do that, you must switch memory allocation to manual mode using the PrlDispCfg_SetAdjustMemAuto (p. 55) function. The amount of currently allocated memory can be retrieved using the PrlDispCfg_GetReservedMemLimit (p. 49) function.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_IsAllowMultiplePMC Function

Determines whether the Allow Multiple PMC option is enabled.

Syntax

```
PRL_RESULT PrlDispCfg_IsAllowMultiplePMC(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_IsBackupUserPasswordEnabled Function

Returns the backup user password enabling sign.

Syntax

```
PRL_RESULT PrlDispCfg_IsBackupUserPasswordEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbUserPasswordEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbUserPasswordEnabled

[out] Pointer to a buffer that receives the backup user password enabling sign

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_IsLogRotationEnabled Function

Determines whether the log rotation is enabled for dispatcher and virtual machine logs.

Syntax

```
PRL_RESULT PrlDispCfg_IsLogRotationEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the log rotation is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PrlDispCfg_IsSendStatisticReport Function

Determines whether send statistics reports (CEP) mechanism activated.

Syntax

```
PRL_RESULT PrlDispCfg_IsSendStatisticReport(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbSendStatisticReport
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbSendStatisticReport

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the mechanism is activated. PRL_FALSE (p. 1626) indicates that mechanism is switched off (reports do not sending).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When this mechanism activated periodically report about Virtuozzo product configuration genrating and sending to the Virtuozzo problem reports server automatically.

See Also

PrlDispCfg_SetSendStatisticReport (p. 69)

PrlDispCfg_IsVerboseLogEnabled Function

Determines whether the verbose log level is configured for dispatcher and virtual machines processes.

Syntax

```
PRL_RESULT PrlDispCfg_IsVerboseLogEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the verbose log level is configured. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PrlDispCfg_SetAdjustMemAuto Function

Allows to choose the Virtuozzo Service memory allocation mode from automatic or manual.

Syntax

```
PRL_RESULT PrlDispCfg_SetAdjustMemAuto(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bAdjustMemAuto
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bAdjustMemAuto

To set the size automatically, set this parameter to PRL_TRUE (p. 1652). To set the size manually, set this parameter to PRL_FALSE (p. 1626) and then use the PrlDispCfg_SetReservedMemLimit (p. 68) function to set the size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, the amount of memory allocated for Virtuozzo Service operations is calculated automatically based on the total RAM available on the host. You can allocate a different amount of memory if needed using the PrlDispCfg_SetReservedMemLimit (p. 68) function. Before you do that, you must switch memory allocation to manual using this function. The current memory allocation method can be determined using the PrlDispCfg_IsAdjustMemAuto (p. 52) function. The amount of currently allocated memory can be retrieved using the PrlDispCfg_GetReservedMemLimit (p. 49) function.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAllowMultiplePMC Function

Enables or disables the Allow Multiple PMC option.

Syntax

```
PRL_RESULT PrlDispCfg_SetAllowMultiplePMC(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bEnabled

[in] Action type. Specify PRL_TRUE (p. 1652) to enable the Allow Multiple PMC option. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetBackupTimeout Function

Set backup timeout in seconds.

Syntax

```
PRL_RESULT PrlDispCfg_SetBackupTimeout(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nTimeout
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nTimeout

[in] timeout in seconds

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetBackupTmpDir Function

Sets the backup temporary directory.

Syntax

```
PRL_RESULT PrlDispCfg_SetBackupTmpDir(
    PRL_HANDLE hDispConfig,
    PRL_CONST_STR sBackupTmpDir
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sBackupTmpDir

[in] The backup temporary directory (a UTF-8 encoded, null-terminated string). Can be null or empty in case when an empty value should be assigned.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetBackupUserLogin Function

Sets the backup user login.

Syntax

```
PRL_RESULT PrlDispCfg_SetBackupUserLogin(
    PRL_HANDLE hDispConfig,
    PRL_CONST_STR sUserLogin
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sUserLogin

[in] The backup user login (a UTF-8 encoded, null-terminated string). Can be null or empty in case when an empty value should be assigned.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetBackupUserPassword Function

Sets the backup user password.

Syntax

```
PRL_RESULT PrlDispCfg_SetBackupUserPassword(
    PRL_HANDLE hDispConfig,
    PRL_CONST_STR sUserPassword
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sUserPassword

[in] The backup user password (a UTF-8 encoded, null-terminated string). Can be null or empty in case when an empty value should be assigned.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetBackupUserPasswordEnabled Function

Sets the backup user password.

Syntax

```
PRL_RESULT PrlDispCfg_SetBackupUserPasswordEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bUserPasswordEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bUserPasswordEnabled

[in] The backup user password enabling sign

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetCanChangeDefaultSettings Function

Allows to grant or deny a permission to new users to modify Virtuozzo Service preferences.

Syntax

```
PRL_RESULT PrlDispCfg_SetCanChangeDefaultSettings(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bDefaultChangeSettings
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bDefaultChangeSettings

If set to PRL_TRUE (p. 1652), new users will be allowed to modify Virtuozzo Service preferences. If set to PRL_FALSE (p. 1626), new users will not be allowed to make such modifications.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, only administrators of the host OS can modify Virtuozzo Service preferences. When a new Virtuozzo Service user profile is created (this happens when a user logs on to the Virtuozzo Service for the first time), he/she will be granted or denied this privilege based on the default setting. This function allows to set that default setting. To determine the current behavior, use PrlDispCfg_CanChangeDefaultSettings (p. 36). Please note that this function only affects the new users (the users that will be created in the future). The profiles of the existing users will not (and cannot) be modified.

See Also

PrlDispCfg_CanChangeDefaultSettings (p. 36)

PrlDispCfg_SetConfirmationsList Function

Obtains a handles list of operations which required administrator confirmation (for all users and for all virtual machines(if operation is applicable to vm))

Syntax

```
PRL_RESULT PrlDispCfg_SetConfirmationsList(
    PRL_HANDLE hDispConfig,
    PRL_HANDLE hConfirmList
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

hConfirmList

variable that receives the handle. The type of the handle is PHT_OPAQUE_TYPE_LIST (p. 177). (values in list consist in enum PRL_ALLOWED_VM_COMMAND (p. 1036))

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a handle.

See Also

PrlSrv_DisableConfirmationMode (p. 236)() PrlSrv_EnableConfirmationMode (p. 236)() PrlSrv_IsConfirmationModeEnabled (p. 268)() PrlVmCfg_GetConfirmationsList (p. 561)() PrlVmCfg_SetConfirmationsList (p. 660)() PrlDispCfg_GetConfirmationsList (p. 39)() PrlDispCfg_SetConfirmationsList()

PrlDispCfg_SetCpuFeaturesMaskEx Function

Allows to set the CPU features masks.

Syntax

```
PRL_RESULT PrlDispCfg_SetCpuFeaturesMaskEx(
    PRL_HANDLE hDispConfig,
    PRL_HANDLE hCpuFeatures
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

hCpuFeatures

A handle of type PHT_CPU_FEATURES (p. 22).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

This masks will be saved at dispatcher config and will be used on VM start.

See Also

PrlDispCfg_GetCpuFeaturesMaskEx (p. 39)

PrlDispCfg_SetDefaultBackupDirectory Function

Sets PATH to the default backup directory.

Syntax

```
PRL_RESULT PrlDispCfg_SetDefaultBackupDirectory(
    PRL_HANDLE hDispConfig,
    PRL_CONST_STR sBackupDirectory
```

```
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sBackupDirectory

[in] PATH to the backup directory (a UTF-8 encoded, null-terminated string). Can be null or empty in case when an empty value should be assigned.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetDefaultBackupServer Function

Sets the default backup server host name or IP address.

Syntax

```
PRL_RESULT PrlDispCfg_SetDefaultBackupServer(
    PRL_HANDLE hDispConfig,
    PRL_CONST_STR sBackupServer
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sBackupServer

[in] The backup server host name or IP address (a UTF-8 encoded, null-terminated string). Can be null or empty in case when an empty value should be assigned.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetDefaultVNCHostName Function

Sets the base VNC host name.

Syntax

```
PRL_RESULT PrlDispCfg_SetDefaultVNCHostName(
    PRL_HANDLE hDispConfig,
    PRL_CONST_STR sNewHostName
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

sNewHostName

The new VNC host name. The value must be UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetLogRotationEnabled Function

Enables or disables the log rotation for dispatcher and virtual machine logs.

Syntax

```
PRL_RESULT PrlDispCfg_SetLogRotationEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bEnabled

[in] Action type. Specify PRL_TRUE (p. 1652) to enable the log rotation. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetMaxReservMemLimit Function

Allows to manually set the upper limit of the memory size that can be reserved for Virtuozzo Service operation.

Syntax

```
PRL_RESULT PrlDispCfg_SetMaxReservMemLimit(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMemSize

The new maximum memory limit (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, memory limits are calculated automatically based on the host machine RAM size. If needed, you can use this function to modify the default value of the maximum reserved memory size. Please note that modifying default memory limits must be done with extreme caution. Unjustified memory limits may result in too much or too little memory reserved for Virtuozzo Service, which in turn may lead to an unstable or inefficient system. While this and other functions allow you to experiment with Virtuozzo Service memory limits, the default values are the ones that should be used in most situations.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_SetMaxVmMem Function

Allows to set the maximum memory size that can be allocated to an individual virtual machine.

Syntax

```
PRL_RESULT PrlDispCfg_SetMaxVmMem(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMemSize

The new memory size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, virtual machine memory limits are calculated automatically. If needed, you can set your own maximum memory limit using this function. Please note that the new value must not exceed the size of the memory reserved for Virtuozzo Service, which can be obtained using the PrlDispCfg_GetReservedMemLimit (p. 49) function. The actual memory size for a virtual machine is set when configuring it and will be validated against the value set here.

See Also

PrlDispCfg_GetMaxVmMem (p. 45)

PrlDispCfg_GetMinVmMem (p. 47)

PrlDispCfg_GetRecommendMaxVmMem (p. 48)

PrlDispCfg_SetMinVmMem (p. 67)

PrlDispCfg_SetRecommendMaxVmMem (p. 68)

PrlDispCfg_SetMinReservMemLimit Function

Allows to manually set the lower limit of the memory size that must be reserved for Virtuozzo Service operation.

Syntax

```
PRL_RESULT PrlDispCfg_SetMinReservMemLimit(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMemSize

The new minimum size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, memory limits are calculated automatically based on the host machine RAM size and other parameters. If needed, you can use this function to modify the default value of the minimum required reserved memory size. Please note that modifying default memory limits must be done with extreme caution. Unjustified memory limits may result in too much or too little memory reserved for Virtuozzo Service, which in turn may lead to an unstable or inefficient system. While this and other functions allow you to experiment with Virtuozzo Service memory limits, the default values are should be used in most situations.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetReservedMemLimit (p. 68)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_SetMinSecurityLevel Function

Allows to set the lowest allowable security level that can be used to connect to the specified Virtuozzo Service.

Syntax

```
PRL_RESULT PrlDispCfg_SetMinSecurityLevel(
    PRL_HANDLE hDispConfig,
    PRL_SECURITY_LEVEL nMinSecurityLevel
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMinSecurityLevel

Security level to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The lowest allowable security level is a setting in the Virtuozzo Service preferences. If a client tries to login using a lower level that the one set by this function, the connection to the Virtuozzo Service will be refused.

See Also

PrlDispCfg_GetMinSecurityLevel (p. 46)

PrlDispCfg_SetMinVmMem Function

Allows to set the minimum required memory size that must be allocated to an individual virtual machine.

Syntax

```
PRL_RESULT PrlDispCfg_SetMinVmMem(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMemSize

The new minimum size (in megabytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, virtual machine memory limits are calculated automatically. If needed, you can set your own minimum memory limit using this function. Please note that the new value must not fall below the size of the minimum memory limit set for Virtuozzo Service, which can be obtained using the PrlDispCfg_GetMinReservMemLimit (p. 45) function. The actual memory size for a virtual machine is set when configuring it and will be validated against the value set here.

See Also

PrlDispCfg_GetMaxVmMem (p. 45)

PrlDispCfg_GetRecommendMaxVmMem (p. 48)

PrlDispCfg_SetMaxVmMem (p. 64)

PrlDispCfg_SetMinVmMem

PrlDispCfg_SetRecommendMaxVmMem (p. 68)

PrlDispCfg_SetRecommendMaxVmMem Function

Allows to manually set recommended memory size for an individual virtual machine.

Syntax

```
PRL_RESULT PrlDispCfg_SetRecommendMaxVmMem(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMemSize

The new recommended memory size in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, the recommended memory size for a virtual machine is calculated automatically. If needed, you can set your own recommended memory size using this function. The value must be selected from the memory limit range that can be determined using the PrlDispCfg_GetMinVmMem (p. 47) and PrlDispCfg_GetMaxVmMem (p. 45) functions. The value by itself is not used as a constraint of any kind and is just a suggestion that the end user may act on or not.

See Also

PrlDispCfg_GetMaxVmMem (p. 45)

PrlDispCfg_GetMinVmMem (p. 47)

PrlDispCfg_GetRecommendMaxVmMem (p. 48)

PrlDispCfg_SetMaxVmMem (p. 64)

PrlDispCfg_SetMinVmMem (p. 67)

PrlDispCfg_SetReservedMemLimit Function

Allows to set the amount of memory that will be allocated for Virtuozzo Service operation.

Syntax

```
PRL_RESULT PrlDispCfg_SetReservedMemLimit(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nMemSize
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nMemSize

The new memory size in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The memory size must fall within the memory limits that can be determined using the PrlDispCfg_GetMaxReservMemLimit (p. 44) and the PrlDispCfg_GetMinReservMemLimit (p. 45) functions. By default, the memory size is calculated automatically based on the total RAM available on the host. If you are changing the default value, you must also switch the Virtuozzo Service memory allocation to manual mode using the PrlDispCfg_SetAdjustMemAuto (p. 55) function.

See Also

PrlDispCfg_GetMaxReservMemLimit (p. 44)

PrlDispCfg_GetMinReservMemLimit (p. 45)

PrlDispCfg_GetReservedMemLimit (p. 49)

PrlDispCfg_SetMaxReservMemLimit (p. 64)

PrlDispCfg_SetMinReservMemLimit (p. 65)

PrlDispCfg_IsAdjustMemAuto (p. 52)

PrlDispCfg_SetAdjustMemAuto (p. 55)

PrlDispCfg_SetSendStatisticReport Function

Allows to switch on/off mechanism of sending statistics reports (CEP).

Syntax

```
PRL_RESULT PrlDispCfg_SetSendStatisticReport(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bSendStatisticReport
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bSendStatisticReport

To switch mechanism on, set this parameter to PRL_TRUE (p. 1652). To switch off mechanism set this parameter to PRL_FALSE (p. 1626).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlDispCfg_IsSendStatisticReport (p. 54)

PrlDispCfg_SetUsbIdentAssociation Function

(Un)Associates USB device with given Vm.

Syntax

```
PRL_RESULT PrlDispCfg_SetUsbIdentAssociation(
    PRL_HANDLE hDispCfg,
    PRL_CONST_STR sSystemName,
    PRL_CONST_STR sVmUuid,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hDispCfg

A handle of type PHT_DISP_CONFIG (p. 32).

sSystemName

System name / id of USB device

sVmUuid

Uuid of Vm to associate USB device with. It could be NULL or empty to clear association.

nFlags

Reserved parameter.

Returns

PRL_RESULT. Posible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid system name was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - insufficient memory to complete operation

PRL_ERR_SUCCESS (p. 1365) - function completed successfully

PrlDispCfg_SetVerboseLogEnabled Function

Enables or disables the verbose log level for dispatcher and virtual machines processes.

Syntax

```
PRL_RESULT PrlDispCfg_SetVerboseLogEnabled(
    PRL_HANDLE hDispConfig,
    PRL_BOOL bEnabled
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

bEnabled

[in] Action type. Specify PRL_TRUE (p. 1652) to enable the verbose debug log level. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetVmCpuLimitType Function

%VM_ONLY%

Syntax

```
PRL_RESULT PrlDispCfg_SetVmCpuLimitType(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nVmCpuLimitType
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nVmCpuLimitType

[in] VM CPU limit type value

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDispCfg_SetVNCBasePort Function

Sets the base VNC port number.

Syntax

```
PRL_RESULT PrlDispCfg_SetVNCBasePort(
    PRL_HANDLE hDispConfig,
    PRL_UINT32 nPort
);
```

File

PrlApiDisp.h

Parameters

hDispConfig

A handle of type PHT_DISP_CONFIG (p. 32).

nPort

Port number to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_EVENT

The PHT_EVENT handle is a container that can contain information about system events or extended error information in asynchronous functions.

Description

When the PHT_EVENT object is used as an event data container, a handle to it is obtained through callback functionality (a handle is passed as a parameter to a callback function). When the object is used as an extended error information container in asynchronous functions, a handle to it is obtained from the PHT_JOB (p. 138) object returned by a function by calling the PrlJob_GetError (p. 140) function.

The common steps in processing an event in a callback function are as follows:

1. Determine the type of the event by calling the PrlEvent_GetType (p. 86) function passing the event handle to it. The event types are defined in the PRL_EVENT_TYPE (p. 1054) enumeration. This will tell you what happened on the Virtuozzo Service side. For example, a virtual machine state has changed, the host configuration has been updated, etc.

2. Determine the event source by calling the PrlEvent_GetIssuerType (p. 83) function. The function returns the event issuer type that you can look up in the PRL_EVENT_ISSUER_TYPE (p. 1053) enumeration. This will tell you if the event was generated by the host, a virtual machine, a web service, etc.
3. If an event was generated by the Virtuozzo Service or a virtual machine, you can obtain the server or the virtual machine handle by using the PrlEvent_GetServer (p. 86) or the PrlEvent_GetVm (p. 87) function respectively. Depending on the event type, you may use a handle to access the Virtuozzo Service or a virtual machine and perform some operations on them if needed.
4. An event object may contain additional parameters, which can be retrieved from it using the PrlEvent_GetParam (p. 84) function. The function returns a handle of type PHT_EVENT_PARAMETER (p. 88) containing an individual parameter data. There may be multiple parameters in a given event object, so the function must be called once for each of the available parameters. To determine the number of parameters, use the PrlEvent_GetParamsCount (p. 85) function. Once you've obtained a PHT_EVENT_PARAMETER (p. 88) handle, you can use its functions to retrieve the parameter values.

Functions

	Name	Description
	PrlEvent_CanBeIgnored (p. 73)	Determines whether the event is critical or not.
	PrlEvent_CreateAnswerEvent (p. 74)	Generates properly formatted and structured answer to a question included in the event.
	PrlEvent_GetDataPtr (p. 80)	Clients can use this method to receive a data buffer specific to the event type.
	PrlEvent_GetErrCode (p. 81)	Returns the code associated with the event.
	PrlEvent_GetErrString (p. 81)	Returns string representation of the event message.
	PrlEvent_GetIssuerId (p. 82)	Returns the event issuer ID (host UUID, virtual machine UUID, etc).
	PrlEvent_GetIssuerType (p. 83)	Returns the event issuer type (virtual machine, host, etc.).
	PrlEvent_GetJob (p. 83)	Obtains the handle of the job that caused the event.
	PrlEvent_GetParam (p. 84)	Obtains a handle of type PHT_EVENT_PARAMETER (p. 88) containing an event parameter data.
	PrlEvent_GetParamByName (p. 85)	Obtains a handle of type PHT_EVENT_PARAMETER (p. 88) containing an event parameter data by specified parameter name.
	PrlEvent_GetParamsCount (p. 85)	Determines the number of the parameters in the event container.
	PrlEvent_GetServer (p. 86)	Obtains a handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service that generated the event.
	PrlEvent_GetType (p. 86)	Returns the type of the event.
	PrlEvent_GetVm (p. 87)	Obtains a handle of type PHT_VIRTUAL_MACHINE (p. 442) referencing the virtual machine that generated the event.
	PrlEvent_IsAnswerRequired (p. 88)	Determines whether the event requires a response from the client.

PrlEvent_CanBeIgnored Function

Determines whether the event is critical or not.

Syntax

```
PRL_RESULT PrlEvent_CanBeIgnored(
    PRL_HANDLE hEvent,
    PRL_BOOL_PTR pbCanBeIgnored
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pbCanBeIgnored

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the event message is critical and should be investigated. PRL_FALSE (p. 1626) indicates that the event message is not critical and can be ignored.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_CreateAnswerEvent Function

Generates properly formatted and structured answer to a question included in the event.

Syntax

```
PRL_RESULT PrlEvent_CreateAnswerEvent(
    PRL_HANDLE hEvent,
    PRL_HANDLE_PTR phEvent,
    PRL_RESULT nAnswer
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

phEvent

[out] A pointer to a variable that receives the result (a handle of type PHT_EVENT (p. 72) containing the answer data).

nAnswer

The code of the answer. The answer object data is assembled based on this value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate event object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

One of the event types that a callback function can receive is PET_DSP_EVT_VM_QUESTION. The event of this type is generated on the Virtuozzo Service side when a user interaction is required. For example, let's say you've sent a request to the Virtuozzo Service to create a virtual machine but specified the size of the virtual hard drive that exceeds the free space available on the physical hard drive. In such a case, the Virtuozzo Service will generate an event containing a question (and possible answers) about how the user would like to proceed. In this particular case, the question will be similar to "The size of the

specified disk is larger than the free space available. Would you like to create the disk anyway?" and the possible answers will be "Yes" or "No". The user must choose one of the available options, compose an answer, and send it to the Virtuozzo Service. Based on the answer received, the Virtuozzo Service will take the appropriate action.

To obtain a string containing the question, use the PrlEvent_GetErrString (p. 81) function. Answer choices are acquired by obtaining a handle of type PHT_EVENT_PARAMETER (p. 88) using the PrlEvent_GetParam (p. 84) function (a single object will contain an individual answer choice). After the user selects an answer, the PrlEvent_CreateAnswerEvent function must be called to populate another object of type PHT_EVENT (p. 72) with the properly formatted answer data. The resulting object must then be passed to the PrlSrv_SendAnswer (p. 294) function which will send the answer to the Virtuozzo Service.

Example

The following is a complete example that demonstrates how to handle events and how to answer to Virtuozzo Service questions. In the example, we create a blank virtual machine and try to add a virtual hard drive to it with the size larger than the free disk space available on the physical drive. This will trigger an event on the Virtuozzo Service side and a question will be sent to the client asking if we really want to create a drive like that. The virtual machine creation operation will not continue unless we send an answer to the Virtuozzo Service.

```
#include "Parallels.h"
#include "Wrappers/SdkWrap/SdkWrap.h"
#include <stdio.h>
#include <stdlib.h>

#ifdef _WIN_
#include <windows.h>
#else
#include <unistd.h>
#endif

#define MY_JOB_TIMEOUT 10000 // Default timeout to use in the PrlJob_Wait function.
#define MY_HDD_SIZE 70*1024 // The size of the new hard drive.
#define MY_STR_BUF_SIZE 1024 // The default buffer size to use for string output.

////////////////////////////////////////////////////////////////////////////

// A helper function that will attempt to crate a hard drive larger
// than the free space available, thus triggering an event on the
// Virtuozzo Service, which will result in Service sending us a question.
static PRL_RESULT create_big_hdd(PRL_HANDLE hVm);

// The callback function (event handler).
static PRL_RESULT callback(PRL_HANDLE, PRL_VOID_PTR);

////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
    // Pick the correct dynamic library file depending on the platform.
    #ifdef _WIN_
        #define SDK_LIB_NAME "prl_sdk.dll"
    #elif defined(_LIN_)
        #define SDK_LIB_NAME "libprl_sdk.so"
    #elif defined(_MAC_)
        #define SDK_LIB_NAME "libprl_sdk.dylib"
    #endif

    // Load the dynamic library.
    if (PRL_FAILED(SdkWrap_Load(SDK_LIB_NAME)) &&
        PRL_FAILED(SdkWrap_Load("./" SDK_LIB_NAME)))
    {
        // Error handling goes here...
        return -1;
    }
```

```
PRL_RESULT ret = PRL_ERR_UNINITIALIZED;
PRL_RESULT err = PRL_ERR_UNINITIALIZED;
PRL_RESULT rc = PRL_ERR_UNINITIALIZED;
PRL_HANDLE hJob = PRL_INVALID_HANDLE;
PRL_HANDLE hJobResult = PRL_INVALID_HANDLE;
PRL_HANDLE hServer = PRL_INVALID_HANDLE;

// Initialize API library.
err = PrlApi_Init(PARALLELS_API_VER);
if (PRL_FAILED(err))
{
    // Error handling goes here...
    return -1;
}

// Create server object.
PrlSrv_Create(&hServer);

// Log in.
hJob = PrlSrv_Login(
    hServer, // Server handle
    "10.30.22.82", // Host IP address
    "jdoe", // User
    "secret", // Password
    0, // Previous session ID
    0, // Port number
    0, // Timeout
    PSL_NORMAL_SECURITY); // Security

ret = PrlJob_Wait(hJob, MY_JOB_TIMEOUT);
PrlHandle_Free(hJob);

if (PRL_FAILED(ret))
{
    fprintf(stderr, "PrlJob_Wait for PrlSrv_Login returned with error: %s\n",
        PRL_RESULT_TO_STRING(ret));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Analyze the result of PrlSrv_Login.
PRL_RESULT nJobResult;
ret = PrlJob_GetRetCode( hJob, &nJobResult );
if (PRL_FAILED( nJobResult))
{
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    printf( "Login job returned with error: %s\n",
        PRL_RESULT_TO_STRING(nJobResult));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Create a new virtual machine.
PRL_HANDLE hVm;
PrlSrv_CreateVm(hServer, &hVm);
PrlVm_SetName(hVm, "My simple VM");

// Register the virtual machine with the Virtuozzo Service.
hJob = PrlVm_Reg(hVm, "", PRL_FALSE);
```

```
    PrlJob_Wait(hJob, MY_JOB_TIMEOUT);
    PrlHandle_Free(hJob);

    // Register the event handler with the Service.
    // The second parameter is a pointer to our callback function.
    PrlSrv_RegEventHandler(hServer, &callback, NULL);

    // Try creating a virtual hard drive larger than the
    // free space available (increase MY_HDD_SIZE value if needed).
    // This should produce an event that will
    // contain a question from the Virtuozzo Service.
    create_big_hdd(hVm);

    //
    // At this point, the background thread should call the
    // callback function.
    //

    // We can now clean up and exit the program.
    // Unregister the event handler and log off.
    PrlSrv_UnregEventHandler(hServer, &callback, NULL);
    hJob = PrlSrv_Logoff(hServer);
    PrlJob_Wait(hJob, MY_JOB_TIMEOUT);
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return 0;
}

////////////////////////////////////////////////////////////////////////////
// The callback function implementation.
// The event handling is demonstrated here.
//
static PRL_RESULT callback(PRL_HANDLE hEvent, PRL_VOID_PTR pUserData)
{
    PRL_HANDLE_TYPE nHandleType;
    PrlHandle_GetType(hEvent, &nHandleType);

    // A callback function will be called more than once.
    // It will be called for every job that we initiate and it
    // will be called for the event that we intentionally trigger.
    // In this example, we are interested in events only.
    if (nHandleType != PHT_EVENT)
    {
        return PrlHandle_Free(hEvent);
    }

    // Get the type of the event received.
    PRL_EVENT_TYPE type;
    PrlEvent_GetType(hEvent, &type);

    // See if the received event is a "question".
    if (type == PET_DSP_EVT_VM_QUESTION)
    {
        PRL_UINT32 nParamsCount = 0;
        PRL_RESULT err = PRL_ERR_UNINITIALIZED;

        // Extract the text of the question and display it on the screen.
        // The question is stored in the object in the "event message" property.
        PRL_BOOL bIsBriefMessage = true;
        char errMsg [MY_STR_BUF_SIZE];
        PRL_UINT32 nBufSize = MY_STR_BUF_SIZE;

        PrlEvent_GetErrString(hEvent, bIsBriefMessage, errMsg, &nBufSize);
        printf("Question: %s\n\n", errMsg);
```

```
// Extract answer choices. They are stored in the
// hEvent object as event parameters.
// First, determine the number of parameters.
err = PrlEvent_GetParamsCount(hEvent, &nParamsCount);
if (PRL_FAILED(err))
{
    fprintf(stderr, "[3]%.8X: %s\n", err,
        PRL_RESULT_TO_STRING(err));
    PrlHandle_Free(hEvent);
    return err;
}

// Declare an array to hold the choices information.
PRL_UINT32_PTR choices =(PRL_UINT32_PTR)
    malloc(nParamsCount * sizeof(PRL_UINT32));

// Now, itereate through the parameter list obtaining a
// handle of type PHT_EVENT_PARAMETER that will contain
// an individual parameter data.
for(PRL_UINT32 nParamIndex = 0; nParamIndex < nParamsCount; ++nParamIndex)
{
    PRL_HANDLE hParam; // this will receive the event parameter handle.
    PRL_RESULT err = PRL_ERR_UNINITIALIZED;

    // The PrlEvent_GetParam function obtains a handle of type
    // PHT_EVENT_PARAMETER containing (in this case) an answer choice.
    err = PrlEvent_GetParam(hEvent, nParamIndex, &hParam);
    if (PRL_FAILED(err))
    {
        fprintf(stderr, "[4]%.8X: %s\n", err,
            PRL_RESULT_TO_STRING(err));
        PrlHandle_Free(hParam);
        PrlHandle_Free(hEvent);
        return err;
    }

    // Get the answer description that can be shown to the user.
    // First, obtain the event parameter value.
    err = PrlEvtPrm_ToUint32(hParam, &choices[nParamIndex]);
    if (PRL_FAILED(err))
    {
        fprintf(stderr, "[9]%.8X: %s\n", err,
            PRL_RESULT_TO_STRING(err));
        PrlHandle_Free(hParam);
        PrlHandle_Free(hEvent);
        return err;
    }

    // Now, get the answer description using the
    // event parameter value as input in the following call.
    char sDesc [MY_STR_BUF_SIZE];
    err = PrlApi_GetResultDescription(choices[nParamIndex], true, sDesc, &nBufSize);
    if (PRL_FAILED(err))
    {
        fprintf(stderr, "[8]%.8X: %s\n", err,
            PRL_RESULT_TO_STRING(err));
        PrlHandle_Free(hParam);
        PrlHandle_Free(hEvent);
        return err;
    }

    // Display the answer choice on the screen.
    printf("Answer choice: %s\n", sDesc);
    PrlHandle_Free(hParam);
}
```

```
        // Select an answer choice (we are using the "No" answer here) and
        // create a valid answer object (hAnswer).
        PRL_HANDLE hAnswer;
        err = PrlEvent_CreateAnswerEvent(hEvent, &hAnswer, choices[1]);
        if (PRL_FAILED(err))
        {
            fprintf(stderr, "[A]%.8X: %s\n", err, PRL_RESULT_TO_STRING(err));
            PrlHandle_Free(hEvent);
            return err;
        }

        // Obtain a server handle.
        PRL_HANDLE hServer;
        PrlEvent_GetServer(hEvent, &hServer);

        // Send the answer handle to the Virtuozzo Service.
        PrlSrv_SendAnswer(hServer, hAnswer);

        free(choices);
        PrlHandle_Free(hServer);
        PrlHandle_Free(hAnswer);
    }
    else // other event type
    {
        PrlHandle_Free(hEvent);
    }

    return PRL_ERR_SUCCESS;
}

////////////////////////////////////////////////////////////////////////////////

// A helper function that will attempt to crate a hard drive larger
// than the free space available, thus triggering an event.
PRL_RESULT create_big_hdd(PRL_HANDLE hVm)
{
    PRL_HANDLE hJobBeginEdit = PRL_INVALID_HANDLE;
    PRL_HANDLE hJobCommit = PRL_INVALID_HANDLE;
    PRL_HANDLE hJob = PRL_INVALID_HANDLE;
    PRL_RESULT nJobRetCode = PRL_ERR_UNINITIALIZED;
    PRL_RESULT err = PRL_ERR_UNINITIALIZED;

    // Timestamp the beginning of the configuration changes operation.
    hJobBeginEdit = PrlVm_BeginEdit(hVm);
    err = PrlJob_Wait(hJobBeginEdit, MY_JOB_TIMEOUT);
    PrlJob_GetRetCode(hJobBeginEdit, &nJobRetCode);
    if (PRL_FAILED(nJobRetCode))
    {
        fprintf(stderr, "[B]%.8X: %s\n", nJobRetCode,
            PRL_RESULT_TO_STRING(nJobRetCode));
        PrlHandle_Free(hJobBeginEdit);
        return nJobRetCode;
    }

    // Create a new device handle.
    // This will be our new virtual hard disk.
    PRL_HANDLE hHDD;
    err = PrlVm_CreateVmDev(
            hVm, // Target virtual machine.
            PDE_HARD_DISK, // Device type.
            &hHDD ); // Device handle.

    // Set disk type to "expanding".
    err = PrlVmDevHd_SetDiskType(hHDD, PHD_EXPANDING_HARD_DISK);
```

```
    // Set max disk size, in megabytes.
    err = PrlVmDevHd_SetDiskSize(hHDD, MY_HDD_SIZE);

    // This option determines whether the image file will be split
    // into chunks or created as a single file.
    err = PrlVmDevHd_SetSplitted(hHDD, PRL_FALSE);

    // Choose and set the name for the new image file.
    err = PrlVmDev_SetFriendlyName(hHDD, "harddisk4.hdd");
    err = PrlVmDev_SetSysName(hHDD, "harddisk4.hdd");

    // Set the emulation type.
    err = PrlVmDev_SetEmulatedType(hHDD, PDT_USE_IMAGE_FILE);

    // Enable the new disk on successful creation.
    err = PrlVmDev_SetEnabled(hHDD, PRL_TRUE);

    // Create the new image file.
    hJob = PrlVmDev_CreateImage(hHDD,
            PRL_TRUE, // Do not overwrite if file exists.
            PRL_FALSE ); // Use non-interactive mode.

    err = PrlJob_Wait(hJob, MY_JOB_TIMEOUT);
    if (PRL_FAILED(err))
    {
        fprintf(stderr, "[C]%.8X: %s\n", err,
            PRL_RESULT_TO_STRING(err));
        PrlHandle_Free(hJob);
        return err;
    }

    // Commit the changes.
    hJobCommit = PrlVm_Commit(hVm);
    err = PrlJob_Wait(hJobCommit, MY_JOB_TIMEOUT);
    PrlJob_GetRetCode(hJobCommit, &nJobRetCode);
    if (PRL_FAILED(nJobRetCode))
    {
        fprintf(stderr, "[D]%.8X: %s\n", nJobRetCode,
            PRL_RESULT_TO_STRING( nJobRetCode));
        PrlHandle_Free(hJobCommit);
        return nJobRetCode;
    }
    return PRL_ERR_SUCCESS;
}
```

PrlEvent_GetDataPtr Function

Clients can use this method to receive a data buffer specific to the event type.

Syntax

```
PRL_RESULT PrlEvent_GetDataPtr(
    PRL_HANDLE hEvent,
    PRL_VOID_PTR_PTR data_ptr
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

data_ptr

[out] A pointer to the event-specific data.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Different event types may provide different data buffers or they may provide none.

PrlEvent_GetErrCode Function

Returns the code associated with the event.

Syntax

```
PRL_RESULT PrlEvent_GetErrCode(
    PRL_HANDLE hEvent,
    PRL_RESULT_PTR pnErrCode
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pnErrCode

[out] A pointer to a variable that receives the code.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetErrString Function

Returns string representation of the event message.

Syntax

```
PRL_RESULT PrlEvent_GetErrString(
    PRL_HANDLE hEvent,
    PRL_BOOL bIsBriefMessage,
    PRL_BOOL bFormated,
    PRL_STR sErrString,
    PRL_UINT32_PTR pnErrStringBufLength
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

bIsBriefMessage

Specifies whether to return a brief or a long version of the message. Specify PBL_TRUE to return a brief message. Specify PRL_FALSE (p. 1626) to return a long message.

bFormated

Specifies whether the HTML tags used for text formatting should be included in the resulting string. PRL_TRUE (p. 1652) - include HTML tags. PRL_FALSE (p. 1626) - don't include.

sErrString

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnErrStringBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetIssuerId Function

Returns the event issuer ID (host UUID, virtual machine UUID, etc).

Syntax

```
PRL_RESULT PrlEvent_GetIssuerId(
    PRL_HANDLE hEvent,
    PRL_STR sIssuerId,
    PRL_UINT32_PTR pnIssuerIdBufLength
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

sIssuerId

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIssuerIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetIssuerType Function

Returns the event issuer type (virtual machine, host, etc.).

Syntax

```
PRL_RESULT PrlEvent_GetIssuerType(
    PRL_HANDLE hEvent,
    PRL_EVENT_ISSUER_TYPE_PTR pnIssuerType
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pnIssuerType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetJob Function

Obtains the handle of the job that caused the event.

Syntax

```
PRL_RESULT PrlEvent_GetJob(
    PRL_HANDLE hEvent,
    PRL_HANDLE_PTR pJob
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pJob

[out] A pointer to a variable that receives a handle of type PHT_JOB (p. 138).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetParam Function

Obtains a handle of type PHT_EVENT_PARAMETER (p. 88) containing an event parameter data.

Syntax

```
PRL_RESULT PrlEvent_GetParam(
    PRL_HANDLE hEvent,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phEventParam
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

nIndex

An index identifying the event parameter (beginning with 0). To determine the number of the parameters in the event container, use the PrlEvent_GetParamsCount (p. 85) function.

phEventParam

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_UNIMPLEMENTED (p. 1373) - the event object does not support this functionality.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

An event container may contain multiple parameters. The meaning of the parameter may be different for different event types.

PrlEvent_GetParamByName Function

Obtains a handle of type PHT_EVENT_PARAMETER (p. 88) containing an event parameter data by specified parameter name.

Syntax

```
PRL_RESULT PrlEvent_GetParamByName(
    PRL_HANDLE hEvent,
    PRL_CONST_STR sParamName,
    PRL_HANDLE_PTR phEventParam
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

sParamName

Event parameter name.

phEventParam

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_UNIMPLEMENTED (p. 1373) - the event object does not support this functionality.

PRL_ERR_NO_DATA (p. 1324) - parameter with specfied name does not exist.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

An event container may contain multiple parameters. The meaning of the parameter may be different for different event types.

PrlEvent_GetParamsCount Function

Determines the number of the parameters in the event container.

Syntax

```
PRL_RESULT PrlEvent_GetParamsCount(
    PRL_HANDLE hEvent,
    PRL_UINT32_PTR pnParamsCount
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pnParamsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_UNIMPLEMENTED (p. 1373) - the event object does not support this functionality.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetServer Function

Obtains a handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service that generated the event.

Syntax

```
PRL_RESULT PrlEvent_GetServer(
    PRL_HANDLE hEvent,
    PRL_HANDLE_PTR server
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

server

[out] A pointer to a variable that receives the server handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When processing an event inside a callback function (event handler) this function is the only way to get the server handle.

PrlEvent_GetType Function

Returns the type of the event.

Syntax

```
PRL_RESULT PrlEvent_GetType(
    PRL_HANDLE hEvent,
    PRL_EVENT_TYPE_PTR type
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

type

[out] A pointer to a variable that receives the result (event type).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvent_GetVm Function

Obtains a handle of type PHT_VIRTUAL_MACHINE (p. 442) referencing the virtual machine that generated the event.

Syntax

```
PRL_RESULT PrlEvent_GetVm(
    PRL_HANDLE hEvent,
    PRL_HANDLE_PTR pVm
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pVm

[out] A pointer to a variable that receives the virtual machine handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When processing an event inside a callback function (event handler) this function is the only way to get the virtual machine

handle.

PrlEvent_IsAnswerRequired Function

Determines whether the event requires a response from the client.

Syntax

```
PRL_RESULT PrlEvent_IsAnswerRequired(
    PRL_HANDLE hEvent,
    PRL_BOOL_PTR pbIsAnswerRequired
);
```

File

PrlApiCore.h

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) identifying the event.

pbIsAnswerRequired

[out] A pointer to a variable that receives the result. PBL_TRUE indicates that a response is required. PBL_TRUE indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

See PrlEvent_CreateAnswerEvent (p. 74) for more information about Virtuozzo Service questions.

PHT_EVENT_PARAMETER

The PHT_EVENT_PARAMETER handle is a container that contains system event parameter data. The handle is obtained by calling the PrlEvent_GetParam (p. 84) function.

Description

The event parameter container contains the following information:

1. Parameter name. To retrieve the name, use the PrlEvtPrm_GetName (p. 90) function.
2. Parameter data type. Depending on the event type, a parameter can be of any type defined in the PRL_PARAM_FIELD_DATA_TYPE_PTR (p. 826) enumeration. To retrieve the parameter data type, use the PrlEvtPrm_GetType (p. 90) function.
3. Parameter value. Depending on the parameter data type, the value must be retrieved using an appropriate function from the PHT_EVENT_PARAMETER group (see the function list below). For example, a boolean value must be retrieved using the PrlEvtPrm_ToBoolean (p. 91) function, the string value must be retrieved using the PrlEvtPrm_ToString (p. 94) function, if a parameter contains a handle, it must be obtained using the PrlEvtPrm_ToHandle (p. 92), etc. The meaning of the value is usually different for different event types.

Functions

	Name	Description
	PrlEvtPrm_GetBuffer (p. 89)	Returns the event parameter value as binary buffer.
	PrlEvtPrm_GetName (p. 90)	Returns the event parameter name.
	PrlEvtPrm_GetType (p. 90)	Returns the event parameter data type.
	PrlEvtPrm_ToBoolean (p. 91)	Returns the event parameter value as boolean (PRL_BOOL (p. 782)).
	PrlEvtPrm_ToCData (p. 91)	Returns the event parameter value as CDATA.
	PrlEvtPrm_ToHandle (p. 92)	Returns the event parameter value as PRL_HANDLE (p. 820).
	PrlEvtPrm_ToInt32 (p. 93)	Returns the event parameter value as PRL_INT32 (p. 792).
	PrlEvtPrm_ToInt64 (p. 93)	Returns the event parameter value as PRL_INT64 (p. 793).
	PrlEvtPrm_ToString (p. 94)	Returns the event parameter value as string.
	PrlEvtPrm_ToUint32 (p. 94)	Returns the event parameter value as PRL_UINT32 (p. 798).
	PrlEvtPrm_ToUint64 (p. 95)	Returns the event parameter value as PRL_UINT64 (p. 799).

PrlEvtPrm_GetBuffer Function

Returns the event parameter value as binary buffer.

Syntax

```
PRL_RESULT PrlEvtPrm_GetBuffer(
    PRL_HANDLE hEventParam,
    PRL_VOID_PTR pParamValue,
    PRL_UINT32_PTR pnParamValueBufLength
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pParamValue

[out] A pointer to a buffer that receives the result.

pnParamValueBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_GetName Function

Returns the event parameter name.

Syntax

```
PRL_RESULT PrlEvtPrm_GetName(
    PRL_HANDLE hEventParam,
    PRL_STR sParamName,
    PRL_UINT32_PTR pnParamNameBufLength
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

sParamName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnParamNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_GetType Function

Returns the event parameter data type.

Syntax

```
PRL_RESULT PrlEvtPrm_GetType(
    PRL_HANDLE hEventParam,
    PRL_PARAM_FIELD_DATA_TYPE_PTR pnParamType
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pnParamType

[out] A pointer to a variable that receives the parameter type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToBoolean Function

Returns the event parameter value as boolean (PRL_BOOL (p. 782)).

Syntax

```
PRL_RESULT PrlEvtPrm_ToBoolean(
    PRL_HANDLE hEventParam,
    PRL_BOOL_PTR pbParamValue
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pbParamValue

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToCData Function

Returns the event parameter value as CDATA.

Syntax

```
PRL_RESULT PrlEvtPrm_ToCData(
    PRL_HANDLE hEventParam,
    PRL_STR sParamValue,
    PRL_UINT32_PTR pnParamValueBufLength
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

sParamValue

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnParamValueBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToHandle Function

Returns the event parameter value as PRL_HANDLE (p. 820).

Syntax

```
PRL_RESULT PrlEvtPrm_ToHandle(
    PRL_HANDLE hEventParam,
    PRL_HANDLE_PTR phParamValue
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

phParamValue

[out] A pointer to a variable that receives the handle. The handle type can be determined using the PrlHandle_GetType (p. 750) function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToInt32 Function

Returns the event parameter value as PRL_INT32 (p. 792).

Syntax

```
PRL_RESULT PrlEvtPrm_ToInt32(
    PRL_HANDLE hEventParam,
    PRL_INT32_PTR pnParamValue
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pnParamValue

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToInt64 Function

Returns the event parameter value as PRL_INT64 (p. 793).

Syntax

```
PRL_RESULT PrlEvtPrm_ToInt64(
    PRL_HANDLE hEventParam,
    PRL_INT64_PTR pnParamValue
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pnParamValue

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToString Function

Returns the event parameter value as string.

Syntax

```
PRL_RESULT PrlEvtPrm_ToString(
    PRL_HANDLE hEventParam,
    PRL_STR sParamValue,
    PRL_UINT32_PTR pnParamValueBufLength
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

sParamValue

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnParamValueBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToUint32 Function

Returns the event parameter value as PRL_UINT32 (p. 798).

Syntax

```
PRL_RESULT PrlEvtPrm_ToUint32(
    PRL_HANDLE hEventParam,
    PRL_UINT32_PTR pnParamValue
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pnParamValue

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlEvtPrm_ToUint64 Function

Returns the event parameter value as PRL_UINT64 (p. 799).

Syntax

```
PRL_RESULT PrlEvtPrm_ToUint64(
    PRL_HANDLE hEventParam,
    PRL_UINT64_PTR pnParamValue
);
```

File

PrlApiCore.h

Parameters

hEventParam

A handle of type PHT_EVENT_PARAMETER (p. 88) identifying the container.

pnParamValue

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_FIREWALL_RULE

The PHT_FIREWALL_RULE handle is used to create and modify firewall rules.

Functions

	Name	Description
	PrlFirewallRule_Create (p. 96)	Creates a new handle of type PHT_FIREWALL_RULE.
	PrlFirewallRule_GetLocalNetAddress (p. 96)	Returns the local address of the specified firewall rule entry.
	PrlFirewallRule_GetLocalPort (p. 97)	Returns the local port.

	PrlFirewallRule_GetProtocol (p. 98)	Returns the protocol of the specified firewall rule entry.
	PrlFirewallRule_GetRemoteNetAddress (p. 98)	Returns the remote address of the specified firewall rule entry.
	PrlFirewallRule_GetRemotePort (p. 99)	Returns the remote port.
	PrlFirewallRule_SetLocalNetAddress (p. 99)	Sets the specified firewall rule entry local address.
	PrlFirewallRule_SetLocalPort (p. 100)	Sets the specified local port.
	PrlFirewallRule_SetProtocol (p. 100)	Sets the specified firewall rule entry protocol.
	PrlFirewallRule_SetRemoteNetAddress (p. 101)	Sets the specified firewall rule entry remote address.
	PrlFirewallRule_SetRemotePort (p. 101)	Sets the specified remote port.

PrlFirewallRule_Create Function

Creates a new handle of type PHT_FIREWALL_RULE (p. 95).

Syntax

```
PRL_RESULT PrlFirewallRule_Create(
    PRL_HANDLE_PTR phRule
);
```

File

PrlApiVm.h

Parameters

phRule

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a port forward handle object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It is the responsibility of the user to free the handle when it is no longer needed.

PrlFirewallRule_GetLocalNetAddress Function

Returns the local address of the specified firewall rule entry.

Syntax

```
PRL_RESULT PrlFirewallRule_GetLocalNetAddress(
    PRL_HANDLE hRule,
    PRL_STR sAddr,
    PRL_UINT32_PTR pnBufLength
```

```
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

sAddr

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_GetLocalPort Function

Returns the local port.

Syntax

```
PRL_RESULT PrlFirewallRule_GetLocalPort(
    PRL_HANDLE hRule,
    PRL_UINT32_PTR pnPort
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

pnPort

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_GetProtocol Function

Returns the protocol of the specified firewall rule entry.

Syntax

```
PRL_RESULT PrlFirewallRule_GetProtocol(
    PRL_HANDLE hRule,
    PRL_STR sProtocol,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

sProtocol

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_GetRemoteNetAddress Function

Returns the remote address of the specified firewall rule entry.

Syntax

```
PRL_RESULT PrlFirewallRule_GetRemoteNetAddress(
    PRL_HANDLE hRule,
    PRL_STR sAddr,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

sAddr

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_GetRemotePort Function

Returns the remote port.

Syntax

```
PRL_RESULT PrlFirewallRule_GetRemotePort(
    PRL_HANDLE hRule,
    PRL_UINT32_PTR pnPort
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

pnPort

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_SetLocalNetAddress Function

Sets the specified firewall rule entry local address.

Syntax

```
PRL_RESULT PrlFirewallRule_SetLocalNetAddress(
    PRL_HANDLE hRule,
    PRL_CONST_STR sAddr
```

```
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

sAddr

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_SetLocalPort Function

Sets the specified local port.

Syntax

```
PRL_RESULT PrlFirewallRule_SetLocalPort(
    PRL_HANDLE hRule,
    PRL_UINT32 nPort
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

nPort

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_SetProtocol Function

Sets the specified firewall rule entry protocol.

Syntax

```
PRL_RESULT PrlFirewallRule_SetProtocol(
```

```
    PRL_HANDLE hRule,
    PRL_CONST_STR sProtocol
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

sProtocol

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_SetRemoteNetAddress Function

Sets the specified firewall rule entry remote address.

Syntax

```
PRL_RESULT PrlFirewallRule_SetRemoteNetAddress(
    PRL_HANDLE hRule,
    PRL_CONST_STR sAddr
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

sAddr

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFirewallRule_SetRemotePort Function

Sets the specified remote port.

Syntax

```
PRL_RESULT PrlFirewallRule_SetRemotePort(
    PRL_HANDLE hRule,
    PRL_UINT32 nPort
);
```

File

PrlApiVm.h

Parameters

hRule

A handle of type PHT_FIREWALL_RULE (p. 95) identifying the firewall rule.

nPort

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_FOUND_VM_INFO

The PHT_FOUND_VM_INFO handle is a container that contains information about a virtual machine as a result of a search operation. The handle is obtained by calling the PrlSrv_StartSearchVms (p. 299) function.

Functions

	Name	Description
	PrlFoundVmInfo_GetConfigPath (p. 102)	Extracts the virtual machine configuration path from the container.
	PrlFoundVmInfo_GetName (p. 103)	Extracts the virtual machine name form the container.
	PrlFoundVmInfo_GetOSVersion (p. 104)	Extracts the virtual machine's OS version from the container.
	PrlFoundVmInfo_IsOldConfig (p. 104)	Determines if the virtual machine has an old configuration.
	PrlFoundVmInfo_IsTemplate (p. 105)	Determines of the specified virtual machine is a template.

PrlFoundVmInfo_GetConfigPath Function

Extracts the virtual machine configuration path from the container.

Syntax

```
PRL_RESULT PrlFoundVmInfo_GetConfigPath(
    PRL_HANDLE handle,
    PRL_STR sPath,
    PRL_UINT32_PTR pnPathBufLength
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_FOUND_VM_INFO (p. 102) identifying the container.

sPath

[out] A pointer to a buffer that receives the path. Pass a null pointer to determine the required buffer size.

pnPathBufLength

[in] The size of the buffer used to receive the output data (in bytes). [out] The required buffer size if the buffer parameter contains a null pointer or if the specified buffer size is not large enough.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFoundVmInfo_GetName Function

Extracts the virtual machine name form the container.

Syntax

```
PRL_RESULT PrlFoundVmInfo_GetName(
    PRL_HANDLE handle,
    PRL_STR sName,
    PRL_UINT32_PTR pnNameBufLength
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_FOUND_VM_INFO (p. 102) identifying the container.

sName

[out] A pointer to a buffer that receives the name. Pass a null pointer to determine the required buffer size.

pnNameBufLength

[in] The size of the buffer used to receive the output data (in bytes). [out] The required buffer size if the buffer parameter contains a null pointer or if the specified buffer size is not large enough.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFoundVmInfo_GetOSVersion Function

Extracts the virtual machine's OS version from the container.

Syntax

```
PRL_RESULT PrlFoundVmInfo_GetOSVersion(
    PRL_HANDLE handle,
    PRL_UINT32_PTR pnOSVersion
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_FOUND_VM_INFO (p. 102) identifying the container.

pnOSVersion

[out] A pointer to a variable that receives the OS version.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFoundVmInfo_IsOldConfig Function

Determines if the virtual machine has an old configuration.

Syntax

```
PRL_RESULT PrlFoundVmInfo_IsOldConfig(
    PRL_HANDLE handle,
    PRL_BOOL_PTR pbOldVersion
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_FOUND_VM_INFO (p. 102) identifying the container.

pbOldVersion

[out] A pointer to a variable that receives the result. A value of "true" indicates that the machine has an old configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFoundVmInfo_IsTemplate Function

Determines of the specified virtual machine is a template.

Syntax

```
PRL_RESULT PrlFoundVmInfo_IsTemplate(
    PRL_HANDLE handle,
    PRL_BOOL_PTR pbIsTemplate
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_FOUND_VM_INFO (p. 102) identifying the container.

pbIsTemplate

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the virtual machine is a template. PRL_TRUE (p. 1652) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_GUEST_OSES_MATRIX

The PHT_GUEST_OSES_MATRIX handle is a container that contains a list of operating system names and versions that are supported on a given server. To obtain the handle, use the PrlSrv_GetSupportedOses (p. 259) function.

Functions

	Name	Description
	PrlOsesMatrix_GetDefaultOsVersion (p. 106)	Returns default guest OS version for specified guest OS type retrieved from remote server (see PrlSrv_GetSupportedOses (p. 259)).
	PrlOsesMatrix_GetSupportedOsesTypes (p. 106)	Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes types list retrieved from remote server (see PrlSrv_GetSupportedOses (p. 259)).

	PrlOsesMatrix_GetSupportedOsesVersions (p. 107)	Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes list for specified guest OS type retrieved from remote server (see PrlSrv_GetSupportedOses (p. 259)).
	PrlOsesMatrix_GetSupportType (p. 108)	Returns the support type for the specified guest OS type retrieved from the remote server (see PrlSrv_GetSupportedOses (p. 259)) and guest OS version (see PrlOsesMatrix_GetSupportedOsesVersions (p. 107))

PrlOsesMatrix_GetDefaultOsVersion Function

Returns default guest OS version for specified guest OS type retrieved from remote server (see PrlSrv_GetSupportedOses (p. 259)).

Syntax

```
PRL_RESULT PrlOsesMatrix_GetDefaultOsVersion(
    PRL_HANDLE hOsesMatrix,
    PRL_UINT8 nGuestOsType,
    PRL_UINT16_PTR pnOsVersion
);
```

File

PrlApiVm.h

Parameters

hOsesMatrix

[in] handle of type PHT_GUEST_OSES_MATRIX (p. 105).

nGuestOsType

[in] requesting guest OS type. In order to determine list of supported guests types please refer PrlOses.h

pnOsVersion

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid jandle, null pointer or unknown OS type was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOsesMatrix_GetSupportedOsesTypes Function

Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes types list retrieved from remote server (see PrlSrv_GetSupportedOses (p. 259)).

Syntax

```
PRL_RESULT PrlOsesMatrix_GetSupportedOsesTypes(
    PRL_HANDLE hOsesMatrix,
    PRL_HANDLE_PTR phOsesTypesList
);
```

File

PrlApiVm.h

Parameters

hOsesMatrix

[in] handle of type PHT_GUEST_OSES_MATRIX (p. 105).

phOsesTypesList

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or invalid handle was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOsesMatrix_GetSupportedOsesVersions Function

Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes list for specified guest OS type retrieved from remote server (see PrlSrv_GetSupportedOses (p. 259)).

Syntax

```
PRL_RESULT PrlOsesMatrix_GetSupportedOsesVersions(
    PRL_HANDLE hOsesMatrix,
    PRL_UINT8 nGuestOsType,
    PRL_HANDLE_PTR phGuestsList
);
```

File

PrlApiVm.h

Parameters

hOsesMatrix

[in] handle of type PHT_GUEST_OSES_MATRIX (p. 105).

nGuestOsType

[in] requesting guest OS type. In order to determine list of supported guests types please refer PrlOses.h

phOsesTypesList

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle, null pointer or unknown OS type was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOsesMatrix_GetSupportType Function

Returns the support type for the specified guest OS type retrieved from the remote server (see PrlSrv_GetSupportedOses (p. 259)) and guest OS version (see PrlOsesMatrix_GetSupportedOsesVersions (p. 107))

Syntax

```
PRL_RESULT PrlOsesMatrix_GetSupportType(
    PRL_HANDLE hOsesMatrix,
    PRL_UINT8 nGuestOsType,
    PRL_UINT16 nGuestOSVersion,
    PRL_GUEST_OS_SUPPORT_TYPE_PTR pnGuestSupportType
);
```

File

PrlApiVm.h

Parameters

hOsesMatrix

[in] handle of type PHT_GUEST_OSES_MATRIX (p. 105).

nGuestOsType

[in] requesting guest OS type. In order to determine the list of supported guest types refer to PrlOses.h

nGuestOSVersion

[in] requesting guest OS version. In order to determine the list of supported guest types refer to PrlOses.h

pnGuestSupportType

[out] A pointer to a variable that receives the result

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - an invalid handle, a null pointer or an unknown OS type/version was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully

PHT_HANDLES_LIST

The PHT_HANDLES_LIST handle is a generic container that can hold a list of other handles. It is typically used to pass a list of handles to a function in a single parameter. The handles in the container can be of the same type or of different types. A new handle is created by making the PrlApi_CreateHandlesList (p. 728) function call.

Functions

	Name	Description
	PrlHndlList_AddItem (p. 109)	Adds the specified handle to the container.
	PrlHndlList_GetItem (p. 109)	Retrieves a handle from the container.
	PrlHndlList_GetItemsCount (p. 110)	Determines the total number of handles in the specified container.
	PrlHndlList_RemoveItem (p. 110)	Removes a handle from the container.

PrlHndlList_AddItem Function

Adds the specified handle to the container.

Syntax

```
PRL_RESULT PrlHndlList_AddItem(
    PRL_HANDLE hHandlesList,
    PRL_HANDLE hItem
);
```

File

PrlApiCore.h

Parameters

hHandlesList

A handle of type PHT_HANDLES_LIST (p. 108) identifying the container.

hItem

A handle to add to the container. No validation of any kind is performed when adding a handle to the list.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

PrlHndlList_GetItem Function

Retrieves a handle from the container.

Syntax

```
PRL_RESULT PrlHndlList_GetItem(
    PRL_HANDLE hHandlesList,
    PRL_UINT32 nItemIndex,
    PRL_HANDLE_PTR phItem
);
```

File

PrlApiCore.h

Parameters

hHandlesList

A handle of type PHT_HANDLES_LIST (p. 108) identifying the container.

nItemIndex

A numeric index of the handle to retrieve (beginning with zero). To get the total number of handles in the container, use the PrlHndlList_GetItemsCount (p. 110) function.

phItem

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or out of range index value was passed.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

PrlHndlList_GetItemsCount Function

Determines the total number of handles in the specified container.

Syntax

```
PRL_RESULT PrlHndlList_GetItemsCount(
    PRL_HANDLE hHandlesList,
    PRL_UINT32_PTR pnItemsCount
);
```

File

PrlApiCore.h

Parameters

hHandlesList

A handle of type PHT_HANDLES_LIST (p. 108) identifying the container.

pnItemsCount

[out] A pointer to a variable that receives the result. Handle count begins with zero.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or out of range index value was passed.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

After determining the count, use the PrlHndlList_GetItem (p. 109) function to retrieve the handle from the container.

PrlHndlList_RemoveItem Function

Removes a handle from the container.

Syntax

```
PRL_RESULT PrlHndlList_RemoveItem(
    PRL_HANDLE hHandlesList,
    PRL_UINT32 nItemIndex
);
```

File

PrlApiCore.h

Parameters

hHandlesList

A handle of type PHT_HANDLES_LIST (p. 108) identifying the container.

nItemIndex

A numeric index of the handle to remove (beginning with zero). To get the total number of handles in the container, use the PrlHndlList_GetItemsCount (p. 110) function.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or out of range index value was passed.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

PHT_HW_GENERIC_DEVICE

PHT_HW_GENERIC_DEVICE is a base handle used to manage physical devices. A physical device can be a real physical device installed on the host computer or, in some cases, a device in a virtual machine as seen by the guest operating system (this handle should not be confused with handles that are used to manage virtual devices).

Description

To obtain a handle identifying a specific device of a particular type, an appropriate function from the PHT_SERVER_CONFIG (p. 314) group must be used. For example, to obtain a handle identifying a floppy disk drive, the PrlSrvCfg_GetFloppyDisk (p. 322) function must be used; to obtain an optical disk drive handle, use the PrlSrvCfg_GetOpticalDisk (p. 333) (other functions are named similarly). Some of these functions obtain a handle of type PHT_HW_GENERIC_DEVICE described here. Others obtain a different handle specific to the device type. These other handles are derived from the PHT_HW_GENERIC_DEVICE handle and contain additional functions. For example, the PHT_HW_HARD_DISK (p. 117) handle is derived from PHT_HW_GENERIC_DEVICE and contains additional functions, such as functions for getting the disk partition information. All of the functions provided by the PHT_HW_GENERIC_DEVICE handle can also be used with derived handles, which means that a derived handle can be passed as a parameter to any of the functions described in this section.

The following is the list of handles that are derived from PHT_HW_GENERIC_DEVICE:

PHT_HW_HARD_DISK (p. 117)

PHT_HW_HARD_DISK_PARTITION (p. 121)

PHT_HW_NET_ADAPTER (p. 126)

PHT_HW_GENERIC_PCI_DEVICE (p. 115)

Functions

	Name	Description
	PrlSrvCfgDev_GetDeviceState (p. 112)	Determines whether a virtual machine can directly use a device, such as a PCI device through IOMMU technology (e.g.
	PrlSrvCfgDev_GetId (p. 112)	Obtains the device ID.
	PrlSrvCfgDev_GetName (p. 113)	Obtains the device name.
	PrlSrvCfgDev_GetType (p. 114)	Obtains the device type.
	PrlSrvCfgDev_IsConnectedToVm (p. 114)	Determines whether the device is connected to a virtual machine.

	PrlSrvCfgDev_SetDeviceState (p. 115)	Allows to specify whether a virtual machine can directly use a device, such as a PCI device through IOMMU technology (e.g.

PrlSrvCfgDev_GetDeviceState Function

Determines whether a virtual machine can directly use a device, such as a PCI device through IOMMU technology (e.g.

Syntax

```
PRL_RESULT PrlSrvCfgDev_GetDeviceState(
    PRL_HANDLE hGenericDevice,
    PRL_GENERIC_DEVICE_STATE_PTR pnDeviceState
);
```

File

PrlApiDisp.h

Parameters

hGenericDevice

A handle of type PHT_HW_GENERIC_DEVICE (p. 111), or one of the descendant types, identifying the device.

pnDeviceState

[out] A pointer to a variable that receives the currently set device access preference.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Intel Vt-d) or a USB device.

See Also

PrlSrvCfgDev_SetDeviceState (p. 115)

PrlSrvCfgDev_GetId Function

Obtains the device ID.

Syntax

```
PRL_RESULT PrlSrvCfgDev_GetId(
    PRL_HANDLE hGenericDevice,
    PRL_STR sDevId,
    PRL_UINT32_PTR pnDevIdBufLength
);
```

File

PrlApiDisp.h

Parameters

hGenericDevice

A handle of type PHT_HW_GENERIC_DEVICE (p. 111), or one of the descendant types, identifying the device.

sDevId

[out] A pointer to a buffer that receives the device ID (a UTF-8 encoded, null-terminated string)

pnDevIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgDev_GetName Function

Obtains the device name.

Syntax

```
PRL_RESULT PrlSrvCfgDev_GetName(
    PRL_HANDLE hGenericDevice,
    PRL_STR sDevName,
    PRL_UINT32_PTR pnDevNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hGenericDevice

A handle of type PHT_HW_GENERIC_DEVICE (p. 111), or one of the descendant types, identifying the device.

sDevName

[out] A pointer to a buffer that receives the device name (a UTF-8 encoded, null-terminated string).

pnDevNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgDev_GetType Function

Obtains the device type.

Syntax

```
PRL_RESULT PrlSrvCfgDev_GetType(
    PRL_HANDLE hGenericDevice,
    PRL_DEVICE_TYPE_PTR pnDevType
);
```

File

PrlApiDisp.h

Parameters

hGenericDevice

A handle of type PHT_HW_GENERIC_DEVICE (p. 111), or one of the descendant types, identifying the device.

pnDevType

[out] A pointer to a variable that receives the device type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgDev_IsConnectedToVm Function

Determines whether the device is connected to a virtual machine.

Syntax

```
PRL_RESULT PrlSrvCfgDev_IsConnectedToVm(
    PRL_HANDLE hGenericDevice,
    PRL_BOOL_PTR pbConnected
);
```

File

PrlApiDisp.h

Parameters

hGenericDevice

A handle of type PHT_HW_GENERIC_DEVICE (p. 111), or one of the descendant types, identifying the device.

pbConnected

[out] A pointer to a variable that receives the device connection status.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgDev_SetDeviceState Function

Allows to specify whether a virtual machine can directly use a device, such as a PCI device through IOMMU technology (e.g.

Syntax

```
PRL_RESULT PrlSrvCfgDev_SetDeviceState(
    PRL_HANDLE hGenericDevice,
    PRL_GENERIC_DEVICE_STATE nDeviceState
);
```

File

PrlApiDisp.h

Parameters

hGenericDevice

A handle of type PHT_HW_GENERIC_DEVICE (p. 111), or one of the descendant types, identifying the device.

pnDeviceState

The device access preference.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Intel Vt-d) or a USB device. When using this function to set a PCI device access, the PrlSrv_ConfigureGenericPci (p. 228) function needs to be called afterwards to apply the changes to the Virtuozzo Service preferences.

See Also

PrlSrvCfgDev_GetDeviceState (p. 112)

PHT_HW_GENERIC_PCI_DEVICE

The PHT_HW_GENERIC_PCI_DEVICE handle is used to obtain information about a generic PCI device installed in the host machine. The handle is obtained by calling the PrlSrvCfg_GetGenericPciDevice (p. 323) function. The handle is derived from PHT_HW_GENERIC_DEVICE (p. 111), which provides additional functions.

Description

Generic PCI devices managed by this handle include devices that can be accessed directly by a virtual machine using the IOMMU technology, such as Intel's Vt-d. If you have a system with a PCI device that support the technology, you can choose the device assignment from one of the following options:

Host Computer. If you select this assignment, the PCI device will be available to the host computer only.

Virtual Machines. If you select this assignment, a pass-through driver will be automatically installed in the host operating

system, which will make the device invisible to it. This will make it possible to assign the device directly to a virtual machine.

To select the PCI device assignment, follows these steps:

1. Call the PrlSrvCfg_GetGenericPciDevicesCount (p. 324) function to determine the number of PCI devices installed in the host.

2. Iterate through the device list calling the PrlSrvCfg_GetGenericPciDevice (p. 323) function in each iteration. The function obtains a handle of type PHT_HW_GENERIC_PCI_DEVICE containing information about an individual device.

3. Call the PrlSrvCfgPci_GetDeviceClass (p. 116) to determine the type of the device (video, network, sound). Choose the device(s) of interest and call the PrlSrvCfgDev_SetDeviceState (p. 115) function for each one passing a value indicating the desired assignment type.

4. Populate the PHT_HANDLES_LIST (p. 108) handle with PHT_HW_GENERIC_PCI_DEVICE handles that you've selected in step 3.

5. Call the PrlSrv_ConfigureGenericPci (p. 228) function passing the PHT_HANDLES_LIST (p. 108) handle as a parameter to submit the changes to the Virtuozzo Service.

6. If you've selected the "Virtual Machines" assignment, you can now add the device to the virtual machine(s) of your choice. Please note that the device can be added to multiple virtual machines but it can be used only by a single machine at any given time. Once the device is added to the virtual machine, the latest device drivers should be installed in it (provided by the device manufacturer) the usual way.

Functions

	Name	Description
	PrlSrvCfgPci_GetDeviceClass (p. 116)	Obtains the class of the PCI device (network, display, sound, etc.).

PrlSrvCfgPci_GetDeviceClass Function

Obtains the class of the PCI device (network, display, sound, etc.).

Syntax

```
PRL_RESULT PrlSrvCfgPci_GetDeviceClass(
    PRL_HANDLE hGenericPci,
    PRL_GENERIC_PCI_DEVICE_CLASS_PTR pnGenericPciClass
);
```

File

PrlApiDisp.h

Parameters

hGenericPci

A valid handle of type PHT_HW_GENERIC_PCI_DEVICE (p. 115) identifying the adapter.

pnGenericPciClass

[out] A pointer to a variable that receives the PCI device class.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_HW_HARD_DISK

The PHT_HW_HARD_DISK handle is used to manage physical hard disks installed on the host computer. The handle is obtained by calling the PrlSrvCfg_GetHardDisk (p. 326) function. The handle is derived from PHT_HW_GENERIC_DEVICE (p. 111), which provides additional functions.

Functions

	Name	Description
	PrlSrvCfgHdd_GetDevId (p. 117)	Returns the hard disk device id.
	PrlSrvCfgHdd_GetDevName (p. 118)	Returns the hard disk device name.
	PrlSrvCfgHdd_GetDevSize (p. 118)	Returns the size of the hard disk device.
	PrlSrvCfgHdd_GetDiskIndex (p. 119)	Returns the index of a hard disk device.
	PrlSrvCfgHdd_GetPart (p. 119)	Obtains a handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the specified hard disk partition.
	PrlSrvCfgHdd_GetPartsCount (p. 120)	Returns the number of partitions available on a hard drive.

PrlSrvCfgHdd_GetDevId Function

Returns the hard disk device id.

Syntax

```
PRL_RESULT PrlSrvCfgHdd_GetDevId(
    PRL_HANDLE hHardDisk,
    PRL_STR sDevId,
    PRL_UINT32_PTR pnDevIdBufLength
);
```

File

PrlApiDisp.h

Parameters

hHardDisk

A handle of type PHT_HW_HARD_DISK (p. 117), identifying the hard disk.

sDevId

[out] Pointer to a buffer that receives the device ID (a UTF-8 encoded, null-terminated string).

pnDevIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to

specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHdd_GetDevName Function

Returns the hard disk device name.

Syntax

```
PRL_RESULT PrlSrvCfgHdd_GetDevName(
    PRL_HANDLE hHardDisk,
    PRL_STR sDevName,
    PRL_UINT32_PTR pnDevNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hHardDisk

A handle of type PHT_HW_HARD_DISK (p. 117), identifying the hard disk.

sDevName

[out] Pointer to a buffer that receives the device name (a UTF-8 encoded, null-terminated string).

pnDevNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHdd_GetDevSize Function

Returns the size of the hard disk device.

Syntax

```
PRL_RESULT PrlSrvCfgHdd_GetDevSize(
    PRL_HANDLE hHardDisk,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiDisp.h

Parameters

hHardDisk

A handle of type PHT_HW_HARD_DISK (p. 117), identifying the hard disk.

pnSize

[out] Pointer to a variable that receives the hard disk size. The size is specified in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHdd_GetDiskIndex Function

Returns the index of a hard disk device.

Syntax

```
PRL_RESULT PrlSrvCfgHdd_GetDiskIndex(
    PRL_HANDLE hHardDisk,
    PRL_UINT32_PTR pnIndex
);
```

File

PrlApiDisp.h

Parameters

hHardDisk

A handle of type PHT_HW_HARD_DISK (p. 117), identifying the hard disk.

pnIndex

[out] Pointer to a variable that receives the hard disk index.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHdd_GetPart Function

Obtains a handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the specified hard disk partition.

Syntax

```
PRL_RESULT PrlSrvCfgHdd_GetPart(
    PRL_HANDLE hHardDisk,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phPart
);
```

File

PrlApiDisp.h

Parameters

hHardDisk

A handle of type PHT_HW_HARD_DISK (p. 117), identifying the hard disk.

nIndex

An index of the hard disk partition for which you would like to obtain a handle. Partitions are index beginning with 0 (zero). The total number of partitions can be obtained by calling the PrlSrvCfgHdd_GetPartsCount (p. 120) function.

phPart

[out] A pointer to a handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the partition.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHdd_GetPartsCount Function

Returns the number of partitions available on a hard drive.

Syntax

```
PRL_RESULT PrlSrvCfgHdd_GetPartsCount(
    PRL_HANDLE hHardDisk,
    PRL_UINT32_PTR pnPartsCount
);
```

File

PrlApiDisp.h

Parameters

hHardDisk

A handle of type PHT_HW_HARD_DISK (p. 117), identifying the hard disk.

pnPartsCount

[out] A pointer to a variable receiving the number of partitions.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function together with the PrlSrvCfgHdd_GetPart (p. 119) function to obtain a handle to an object containing a hard disk partition information.

PHT_HW_HARD_DISK_PARTITION

The PHT_HW_HARD_DISK_PARTITION handle is used to manage partitions of a physical hard disk installed on the host computer. The handle is obtained by calling the PrlSrvCfgHdd_GetPart (p. 119) function. The handle is derived from PHT_HW_GENERIC_DEVICEt, which provides additional functions.

Functions

	Name	Description
	PrlSrvCfgHddPart_GetIndex (p. 121)	Returns the index of the specified hard disk partition.
	PrlSrvCfgHddPart_GetName (p. 122)	Returns a hard disk partition name.
	PrlSrvCfgHddPart_GetSize (p. 122)	Returns the hard disk partition size.
	PrlSrvCfgHddPart_GetSysName (p. 123)	Returns a hard disk partition system name.
	PrlSrvCfgHddPart_GetType (p. 124)	Returns a numerical code identifying the type of the specified partition.
	PrlSrvCfgHddPart_IsActive (p. 124)	Determines whether a disk partition is active or inactive.
	PrlSrvCfgHddPart_IsInUse (p. 125)	Determines whether a disk partition is in use, that is, whether it contains a valid file system, being used for swap or by a database, etc.
	PrlSrvCfgHddPart_IsLogical (p. 125)	Determines whether the specified partition is a logical partition.

PrlSrvCfgHddPart_GetIndex Function

Returns the index of the specified hard disk partition.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_GetIndex(
    PRL_HANDLE hPart,
    PRL_UINT32_PTR pnIndex
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

pnIndex

[out] Pointer to a variable that receives the partition index.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Partition index uniquely identifies a partition on a hard disk.

PrlSrvCfgHddPart_GetName Function

Returns a hard disk partition name.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_GetName(
    PRL_HANDLE hPart,
    PRL_STR sName,
    PRL_UINT32_PTR pnNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

sName

[out] Pointer to a buffer that receives the partition name (a UTF-8 encoded, null-terminated string).

pnNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHddPart_GetSize Function

Returns the hard disk partition size.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_GetSize(
    PRL_HANDLE hPart,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

pnSize

[out] Pointer to a buffer that receives the partition size. The size is specified in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHddPart_GetSysName Function

Returns a hard disk partition system name.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_GetSysName(
    PRL_HANDLE hPart,
    PRL_STR sSysName,
    PRL_UINT32_PTR pnSysNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hPart

A handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

sSysName

[out] Pointer to a buffer that receives the partition name (a UTF-8 encoded, null-terminated string).

pnSysNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHddPart_GetType Function

Returns a numerical code identifying the type of the specified partition.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_GetType(
    PRL_HANDLE hPart,
    PRL_UINT32_PTR pnType
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

pnType

[out] Pointer to a variable that receives a numerical code identifying the partition type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHddPart_IsActive Function

Determines whether a disk partition is active or inactive.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_IsActive(
    PRL_HANDLE hPart,
    PRL_BOOL_PTR pbActive
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

pbActive

[out] Pointer to a variable that receives the return value. PRL_TRUE (p. 1652) indicates that the partition is active. PRL_FALSE (p. 1626) indicates that the partition is inactive.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHddPart_IsInUse Function

Determines whether a disk partition is in use, that is, whether it contains a valid file system, being used for swap or by a database, etc.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_IsInUse(
    PRL_HANDLE hPart,
    PRL_BOOL_PTR pbInUse
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

pbInUse

[out] Pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the partition is in use. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgHddPart_IsLogical Function

Determines whether the specified partition is a logical partition.

Syntax

```
PRL_RESULT PrlSrvCfgHddPart_IsLogical(
    PRL_HANDLE hPart,
    PRL_BOOL_PTR pbLogical
);
```

File

PrlApiDisp.h

Parameters

hPart

A valid handle of type PHT_HW_HARD_DISK_PARTITION (p. 121) identifying the hard disk partition.

pbLogical

[out] Pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that this is a logical partition. PRL_FALSE (p.

1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_HW_NET_ADAPTER

The PHT_HW_NET_ADAPTER handle is used to manage network adapters installed on the host computer or in a guest operating system. The handle is obtained by calling the PrlSrvCfg_GetNetAdapter (p. 332) function. The handle is derived from the PHT_HW_GENERIC_DEVICE (p. 111) handle, which provides additional functions.

Functions

	Name	Description
	PrlSrvCfgNet_GetDefaultGateway (p. 126)	Returns the default gateway address for the specified network adapter.
	PrlSrvCfgNet_GetDefaultGatewayIPv6 (p. 127)	Returns the default IPv6 gateway address for the specified network adapter.
	PrlSrvCfgNet_GetDnsServers (p. 128)	Obtains a list of addresses of DNS servers assigned to the specified network adapter.
	PrlSrvCfgNet_GetMacAddress (p. 128)	Returns the MAC address of the specified network adapter.
	PrlSrvCfgNet_GetNetAdapterType (p. 129)	Returns a network adapter type.
	PrlSrvCfgNet_GetNetAddresses (p. 129)	Obtains a list of network addresses (IP address/Subnet mask pairs) assigned to the specified network adapter.
	PrlSrvCfgNet_GetSearchDomains (p. 130)	Obtains a list of search domains assigned to the specified network adapter.
	PrlSrvCfgNet_GetSysIndex (p. 131)	Returns a network adapter system index.
	PrlSrvCfgNet_GetVlanTag (p. 131)	Returns the VLAN tag of the specified network adapter.
	PrlSrvCfgNet_IsConfigureWithDhcp (p. 132)	Determines whether the adapter network settings are configured through DHCP.
	PrlSrvCfgNet_IsConfigureWithDhcpIPv6 (p. 132)	Determines whether the adapter IPv6 network settings are configured through DHCP.
	PrlSrvCfgNet_IsEnabled (p. 133)	Determines whether the adapter is enabled or disabled.

PrlSrvCfgNet_GetDefaultGateway Function

Returns the default gateway address for the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetDefaultGateway(
    PRL_HANDLE hNetAdapter,
    PRL_STR sDefaultGateway,
    PRL_UINT32_PTR pnDefaultGatewayBufLength
```

```
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the network adapter.

sDefaultGateway

[out] Pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetDefaultGatewayIPv6 Function

Returns the default IPv6 gateway address for the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetDefaultGatewayIPv6(
    PRL_HANDLE hNetAdapter,
    PRL_STR sDefaultGateway,
    PRL_UINT32_PTR pnDefaultGatewayBufLength
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the network adapter.

sDefaultGateway

[out] Pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetDnsServers Function

Obtains a list of addresses of DNS servers assigned to the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetDnsServers(
    PRL_HANDLE hNetAdapter,
    PRL_HANDLE_PTR phDnsServersList
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

phDnsServersList

[out] A pointer to a variable that receives the PHT_STRINGS_LIST (p. 357) handle containing the list of addresses.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate strings list object

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetMacAddress Function

Returns the MAC address of the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetMacAddress(
    PRL_HANDLE hNetAdapter,
    PRL_STR sMacAddress,
    PRL_UINT32_PTR pnMacAddressBufLength
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the network adapter.

sMacAddress

[out] Pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnMacAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetNetAdapterType Function

Returns a network adapter type.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetNetAdapterType(
    PRL_HANDLE hNetAdapter,
    PRL_HW_INFO_NET_ADAPTER_TYPE_PTR pnNetAdapterType
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

pnNetAdapterType

[out] A pointer to a variable that receives the network adapter type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetNetAddresses Function

Obtains a list of network addresses (IP address/Subnet mask pairs) assigned to the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetNetAddresses(
    PRL_HANDLE hNetAdapter,
    PRL_HANDLE_PTR phNetAddressesList
```

```
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

phNetAddressesList

[out] A pointer to a variable that receives the PHT_STRINGS_LIST (p. 357) handle containing the addresses.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate strings list object

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetSearchDomains Function

Obtains a list of search domains assigned to the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetSearchDomains(
    PRL_HANDLE hNetAdapter,
    PRL_HANDLE_PTR phSearchDomainsList
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

phSearchDomainsList

[out] A pointer to a variable that receives the PHT_STRINGS_LIST (p. 357) handle containing the list of search domains.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate strings list object

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_GetSysIndex Function

Returns a network adapter system index.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetSysIndex(
    PRL_HANDLE hNetAdapter,
    PRL_UINT32_PTR pnSysIndex
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

pnSysIndex

[out] A pointer to a variable that receives the adapter index.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A system index uniquely identifies an adapter on a physical host.

PrlSrvCfgNet_GetVlanTag Function

Returns the VLAN tag of the specified network adapter.

Syntax

```
PRL_RESULT PrlSrvCfgNet_GetVlanTag(
    PRL_HANDLE hNetAdapter,
    PRL_UINT16_PTR pnVlanTag
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the network adapter.

pnVlanTag

[out] The buffer for storing result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Since the same physical adapter can be used at several virtual networks pair can identify adapter.

PrlSrvCfgNet_IsConfigureWithDhcp Function

Determines whether the adapter network settings are configured through DHCP.

Syntax

```
PRL_RESULT PrlSrvCfgNet_IsConfigureWithDhcp(
    PRL_HANDLE hNetAdapter,
    PRL_BOOL_PTR pbConfigureWithDhcp
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

pbEnabled

[out] A pointer to a variable that receives the return value. PRL_TRUE (p. 1652) indicates that the adapter settings are configured through DHCP. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_IsConfigureWithDhcpIPv6 Function

Determines whether the adapter IPv6 network settings are configured through DHCP.

Syntax

```
PRL_RESULT PrlSrvCfgNet_IsConfigureWithDhcpIPv6(
    PRL_HANDLE hNetAdapter,
    PRL_BOOL_PTR pbConfigureWithDhcp
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

pbEnabled

[out] A pointer to a variable that receives the return value. PRL_TRUE (p. 1652) indicates that the adapter settings are configured through DHCP. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgNet_IsEnabled Function

Determines whether the adapter is enabled or disabled.

Syntax

```
PRL_RESULT PrlSrvCfgNet_IsEnabled(
    PRL_HANDLE hNetAdapter,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiDisp.h

Parameters

hNetAdapter

A valid handle of type PHT_HW_NET_ADAPTER (p. 126) identifying the adapter.

pbEnabled

[out] A pointer to a variable that receives the return value. PRL_TRUE (p. 1652) indicates that the adapter is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_IPPRIV_NET

The PHT_IPPRIV_NET handle is used to create and configure IP private networks.

Functions

	Name	Description
	PrlIPPrivNet_Create (p. 134)	Creates a new handle of type PHT_IPPRIV_NET.
	PrlIPPrivNet_GetName (p. 134)	Returns the name of the specified private network.
	PrlIPPrivNet_GetNetAddresses (p. 135)	Obtains a list of IP addresses in the address/subnet_mask format which is assigned to a private network.
	PrlIPPrivNet_IsGlobal (p. 136)	Determines the mode (global or not) of the specified private network.
	PrlIPPrivNet_SetGlobal (p. 136)	Sets the mode (global or not) of the specified private network.
	PrlIPPrivNet_SetName (p. 137)	Sets the specified private network ID.
	PrlIPPrivNet_SetNetAddresses (p. 137)	Allows to set IP addresses/Subnet masks to a private network.

PrlIPPrivNet_Create Function

Creates a new handle of type PHT_IPPRIV_NET (p. 133).

Syntax

```
PRL_RESULT PrlIPPrivNet_Create(
    PRL_HANDLE_PTR phPrivNet
);
```

File

PrlApiNet.h

Parameters

phPrivNet

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to create and configure an IP private network. If you would like to create a new IP private network, use this function to create the handle first and then use the handle functions to configure the network. If you would like to modify the configuration of an existing network, use the PrlSrv_GetIPPrivateNetworksList (p. 248) function to obtain a handle to the private network of interest.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlIPPrivNet_GetName Function

Returns the name of the specified private network.

Syntax

```
PRL_RESULT PrlIPPrivNet_GetName(
    PRL_HANDLE hPrivNet,
    PRL_STR sName,
    PRL_UINT32_PTR pnNameBufLength
);
```

File

PrlApiNet.h

Parameters

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network.

sName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlIPPrivNet_GetNetAddresses Function

Obtains a list of IP addresses in the address/subnet_mask format which is assigned to a private network.

Syntax

```
PRL_RESULT PrlIPPrivNet_GetNetAddresses(
    PRL_HANDLE hPrivNet,
    PRL_HANDLE_PTR pNetAddresses
);
```

File

PrlApiNet.h

Parameters

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network.

phNetAddressesList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle containing the IP addresses.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate strings list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Returns the list of pairs "IP address/Subnet mask" which is applied to the private network. The result will be returned as list of strings (PHT_STRINGS_LIST (p. 357) type) of pairs "IP address/Subnet mask".

PrlIPPrivNet_IsGlobal Function

Determines the mode (global or not) of the specified private network.

Syntax

```
PRL_RESULT PrlIPPrivNet_IsGlobal(
    PRL_HANDLE hPrivNet,
    PRL_BOOL_PTR pbGlobal
);
```

File

PrlApiNet.h

Parameters

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network.

pbGlobal

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlIPPrivNet_SetGlobal Function

Sets the mode (global or not) of the specified private network.

Syntax

```
PRL_RESULT PrlIPPrivNet_SetGlobal(
    PRL_HANDLE hPrivNet,
    PRL_BOOL bGlobal
);
```

File

PrlApiNet.h

Parameters

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network.

bGlobal

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlIPPrivNet_SetName Function

Sets the specified private network ID.

Syntax

```
PRL_RESULT PrlIPPrivNet_SetName(
    PRL_HANDLE hPrivNet,
    PRL_CONST_STR sName
);
```

File

PrlApiNet.h

Parameters

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network.

sNetworkId

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlIPPrivNet_SetNetAddresses Function

Allows to set IP addresses/Subnet masks to a private network.

Syntax

```
PRL_RESULT PrlIPPrivNet_SetNetAddresses(
    PRL_HANDLE hPrivNet,
    PRL_HANDLE NetAddresses
);
```

File

PrlApiNet.h

Parameters

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network.

hNetAddressesList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of "IP address/Subnet mask" pairs to be applied to the network adapter. Can be PRL_INVALID_HANDLE (p. 1627) if it's necessary to specify an empty list.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network adapter specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_JOB

The PHT_JOB handle provides methods for managing asynchronous operations.

Description

Every asynchronous function in the Virtuozzo API returns a handle of type PHT_JOB. An asynchronous function returns immediately and the corresponding task is processed in a background thread. The returned job object is used as a reference to the job. While the operation is running, the job object can be used to find out the status of the operation, to get a progress report, or to cancel the job if needed. When the job completes, the job object is populated with the operation return code and the operation result (where applicable). The methods provided by the object can be used to extract this information. Once you are done processing the results, you must destroy the job object using the PrlHandle_Free (p. 748) function. For detailed information about jobs and asynchronous functions, please also see the **Virtuozzo Virtualization SDK Programmer's Guide**.

Functions

	Name	Description
	PrlJob_Cancel (p. 139)	Cancel the specified job.
	PrlJob_GetDataPtr (p. 139)	The PrlJob_GetDataPtr function is used to obtain a pointer to a buffer containing the results of an asynchronous operation.
	PrlJob_GetError (p. 140)	Provides additional job error information.
	PrlJob_GetOpCode (p. 141)	Returns the job operation code.
	PrlJob_GetProgress (p. 141)	Obtains a job progress report.
	PrlJob_GetResult (p. 142)	Obtains a handle of type PHT_RESULT (p. 210) containing the result returned by the job.
	PrlJob_GetRetCode (p. 143)	Obtains the return code from the specified job object.
	PrlJob_GetStatus (p. 143)	Obtains the current status of the specified job.
	PrlJob_IsRequestWasSent (p. 144)	Returns the sign whether request was sent to remote side or was aborted by some local reasons (wrong params passed or something else).
	PrlJob_Wait (p. 144)	The PrlJob_Wait function allows to use asynchronous functions synchronously.

PrlJob_Cancel Function

Cancel the specified job.

Syntax

```
PRL_HANDLE PrlJob_Cancel(
    PRL_HANDLE hJob
);
```

File

PrlApiCore.h

Parameters

hJob

A handle of type PHT_JOB (p. 138) identifying the job.

Returns

PRL_HANDLE (p. 820) to the job being canceled.

PRL_INVALID_HANDLE (p. 1627) - not enough memory to instantiate asynchronous operation.

The results are retrieved using the PrlJob_GetRetCode (p. 143) function. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_JOB (p. 138) handle was passed.

PRL_ERR_UNINITIALIZED (p. 1373) - job instance was not bound to Virtuozzo Service connection.

Remarks

If an asynchronous operation takes a long time to complete, you may cancel it by calling this function. The call does not destroy the job object.

PrlJob_GetDataPtr Function

The PrlJob_GetDataPtr function is used to obtain a pointer to a buffer containing the results of an asynchronous operation.

Syntax

```
PRL_RESULT PrlJob_GetDataPtr(
    PRL_HANDLE hJob,
    PRL_VOID_PTR data_ptr,
    PRL_UINT32_PTR data_size
);
```

File

PrlApiCore.h

Parameters

data_ptr

[out] A pointer to a buffer that receives the data. Pass a null pointer to determine the required buffer size.

data_size

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

handle

A handle of type PHT_JOB (p. 138) identifying the job.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Different operations may or may not provide a data buffer as a result of their execution. This function should be used only when an operation results in a buffer populated with the data specific to that operation.

PrlJob_GetError Function

Provides additional job error information.

Syntax

```
PRL_RESULT PrlJob_GetError(
    PRL_HANDLE hJob,
    PRL_HANDLE_PTR phError
);
```

File

PrlApiCore.h

Parameters

hJob

A handle of type PHT_JOB (p. 138) identifying the job.

phError

[out] A pointer to a variable that receives a handle of type PHT_EVENT (p. 72) containing the error information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no additional error information is available.

PRL_ERR_UNIMPLEMENTED (p. 1373) - the functionality is not available for this job type.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If a job returned an error code (see PrlJob_GetRetCode (p. 143)), you may call this function to see if additional error information is available. If the information is available, the function returns with PRL_ERR_SUCCESS (p. 1365) and obtains a handle of type PHT_EVENT (p. 72) containing the information. To extract the information from the object, use the PrlEvent_GetErrString (p. 81) function.

PrlJob_GetOpCode Function

Returns the job operation code.

Syntax

```
PRL_RESULT PrlJob_GetOpCode(
    PRL_HANDLE hJob,
    PRL_JOB_OPERATION_CODE_PTR pnOpCode
);
```

File

PrlApiCore.h

Parameters

pnOpCode

[out] A pointer to a variable that receives the result.

handle

A handle of type PHT_JOB (p. 138) identifying the job.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A job operation code can be used to identify the function that started the corresponding operation and to determine the type of the result the job object contains. Use this function in an event handler (callback function) when examining a received job object. For example, the operation code PJOC_SRV_LOGIN means that this job belongs to login operation (PrlSrv_Login (p. 272), PrlSrv_LoginLocal (p. 276)); the PJOC_SRV_GET_VM_LIST code means that the job was started by the PrlSrv_GetVmList (p. 264) function call and contains a list of virtual machines, etc.

PrlJob_GetProgress Function

Obtains a job progress report.

Syntax

```
PRL_RESULT PrlJob_GetProgress(
    PRL_HANDLE hJob,
    PRL_UINT32_PTR percentage
);
```

File

PrlApiCore.h

Parameters

percentage

[out] A pointer to a variable that receives the result (an integer value from 0 to 100).

handle

valid PRL_HANDLE (p. 820)

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The job progress is reported as percentage of completion.

PrlJob_GetResult Function

Obtains a handle of type PHT_RESULT (p. 210) containing the result returned by the job.

Syntax

```
PRL_RESULT PrlJob_GetResult(
    PRL_HANDLE hJob,
    PRL_HANDLE_PTR handle
);
```

File

PrlApiCore.h

Parameters

hJob

A handle of type PHT_JOB (p. 138) identifying the job.

handle

[out] pointer to the job result handle

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OPERATION_PENDING (p. 1333) - the asynchronous operation that this job refers to is still in progress.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Depending on the job, the PHT_RESULT (p. 210) object may contain a handle, a list of handles, a string, or a list of strings. The description of each asynchronous function in this documentation contains information whether the function returns a PHT_RESULT (p. 210) object or not, and provides instructions on how to obtain it.

Example

The following code illustrates how to obtain a result from the job object using the login operation as an example.

```
// Log in to a Virtuozzo Service.
hJob =
PrlSrv_LoginLocal( hServer, NULL, 0, PSL_LOW_SECURITY );

// Wait for the job to complete.
PrlJob_Wait( hJob, 10000 );
```

```
// Get a handle of type PHT_RESULT containing the results
// of the operation.
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_LOGIN_RESPONSE handle from the job result object.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlJob_GetRetCode Function

Obtains the return code from the specified job object.

Syntax

```
PRL_RESULT PrlJob_GetRetCode(
    PRL_HANDLE hJob,
    PRL_RESULT_PTR retcode
);
```

File

PrlApiCore.h

Parameters

hJob

A handle of type PHT_JOB (p. 138) identifying the job.

retcode

[out] A pointer to a variable that receives the return code.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

On asynchronous operation completion, the job object will contain a return code indicating whether the operation succeeded or failed. Different jobs may return different error codes. The most common codes are: PRL_ERR_INVALID_ARG (p. 1290) (invalid input parameters were specified in the asynchronous function call) and PRL_ERR_SUCCESS (p. 1365) (function successfully completed).

PrlJob_GetStatus Function

Obtains the current status of the specified job.

Syntax

```
PRL_RESULT PrlJob_GetStatus(
    PRL_HANDLE hJob,
    PRL_JOB_STATUS_PTR status
);
```

File

PrlApiCore.h

Parameters

hJob

A handle if type PHT_JOB (p. 138) identifying the job.

status

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlJob_IsRequestWasSent Function

Returns the sign whether request was sent to remote side or was aborted by some local reasons (wrong params passed or something else).

Syntax

```
PRL_RESULT PrlJob_IsRequestWasSent(
    PRL_HANDLE hJob,
    PRL_BOOL_PTR pbIsRequestWasSent
);
```

File

PrlApiCore.h

Parameters

handle

A handle of type PHT_JOB (p. 138) identifying the job.

pbIsOwner

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlJob_Wait Function

The PrlJob_Wait function allows to use asynchronous functions synchronously.

Syntax

```
PRL_RESULT PrlJob_Wait(
    PRL_HANDLE hJob,
    PRL_UINT32 msecs
);
```

File

PrlApiCore.h

Parameters

hJob

A handle of type PHT_JOB (p. 138) identifying the job.

msecs

Timeout in milliseconds. The timeout must be long enough for the job to complete. The optimal value is determined empirically. For an infinite timeout, use the UINT_MAX value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - an invalid PHT_JOB (p. 138) handle was passed.

PRL_ERR_TIMEOUT (p. 1370) - the specified timeout limit was reached.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

The function should be called immediately after the corresponding asynchronous function call. It suspends the main thread for the specified number of milliseconds and waits for the job to complete. If the job completes before the timeout value is reached, the function returns and the execution of the main thread continues normally. If the function times out, it returns an error. On function return, the job object will not be destroyed and could be queried until its destruction with the call to PrlHandle_Free (p. 748).

Example

```
// Log in to Virtuozzo Service.
// This is an asynchronous call.
// Returns a handle to the Job object.
PRL_HANDLE hJob = PrlSrv_Login(
    hServer, "10.30.23.105", "miket", "1q2w3e",
    0, 0, 0, PRL_SECURITY_LEVEL::PSL_LOW_SECURITY);

// Wait for the Job object to receive the results of the operation.
ret = PrlJob_Wait( hJob, 1000 );
if (PRL_FAILED(ret))
{
    printf("PrlJob_Wait returned with error (%s)\n",
        PRL_RESULT_TO_STRING(ret));
    return -1;
}

// Get the results from the Job object.
PRL_RESULT nJobResult;
ret = PrlJob_GetRetCode( hJob, &nJobResult );
if (PRL_FAILED( nJobResult ))
{
    printf("login job returned with error (%s)\n",
        PRL_RESULT_TO_STRING( nJobResult ));
    return -1;
}
else
```

```
    printf( "Login succeeded.\n" );

// Delete the Job handle.
PrlHandle_Free( hJob );
```

PHT_LICENSE

The PHT_LICENSE handle provides methods for retrieving the Virtuozzo license information. The handle is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetLicenseInfo (p. 249) function.

Functions

	Name	Description
	PrlLic_GetCompanyName (p. 146)	Returns the name of the company to which this license was issued.
	PrlLic_GetLicenseKey (p. 147)	Returns the license key string.
	PrlLic_GetStatus (p. 147)	Returns the license status.
	PrlLic_GetUserName (p. 148)	Returns the license user name.
	PrlLic_IsValid (p. 149)	Determines if the license is valid.

PrlLic_GetCompanyName Function

Returns the name of the company to which this license was issued.

Syntax

```
PRL_RESULT PrlLic_GetCompanyName(
    PRL_HANDLE hLicense,
    PRL_STR sCompanyName,
    PRL_UINT32_PTR pnCompanyNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hLicense

A handle of type PHT_LICENSE (p. 146) identifying the license.

sCompanyName

[out] A pointer to a buffer that receives the name (a UTF-8 encoded, null-terminated string).

pnCompanyNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the specified license object does not contain license data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLic_GetLicenseKey Function

Returns the license key string.

Syntax

```
PRL_RESULT PrlLic_GetLicenseKey(
    PRL_HANDLE hLicense,
    PRL_STR sLicenseKey,
    PRL_UINT32_PTR pnLicenseKeyBufLength
);
```

File

PrlApiDisp.h

Parameters

hLicense

A handle of type PHT_LICENSE (p. 146) identifying the license.

sLicenseKey

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnLicenseKeyBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the specified license object does not contain license data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLic_GetStatus Function

Returns the license status.

Syntax

```
PRL_RESULT PrlLic_GetStatus(
    PRL_HANDLE hLicense,
    PRL_RESULT_PTR pnValue
);
```

File

PrlApiDisp.h

Parameters

hLicense

A handle of type PHT_LICENSE (p. 146) identifying the license.

pnValue

[out] A pointer to a variable that receives the license status.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the specified license object does not contain license data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLic_GetUserName Function

Returns the license user name.

Syntax

```
PRL_RESULT PrlLic_GetUserName(
    PRL_HANDLE hLicense,
    PRL_STR sUserName,
    PRL_UINT32_PTR pnUserNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hLicense

A handle of type PHT_LICENSE (p. 146) identifying the license.

sUserName

[out] A pointer to a buffer that receives the user name (a UTF-8 encoded, null-terminated string).

pnUserNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the specified license object does not contain license data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLic_IsValid Function

Determines if the license is valid.

Syntax

```
PRL_RESULT PrlLic_IsValid(
    PRL_HANDLE hLicense,
    PRL_BOOL_PTR pbValue
);
```

File

PrlApiDisp.h

Parameters

hLicense

A handle of type PHT_LICENSE (p. 146) identifying the license.

pbValue

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the license is valid. PRL_FALSE (p. 1626) indicates that the license is invalid.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the specified license object does not contain license data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_LOGIN_RESPONSE

The PHT_LOGIN_RESPONSE handle is a container that contains information pertinent to a successful Virtuozzo Service login operation. The handle is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_Login (p. 272) or the PrlSrv_LoginLocal (p. 276) function.

Functions

	Name	Description
	PrlLoginResponse_GetHostOsVersion (p. 150)	Returns the host OS version.
	PrlLoginResponse_GetProductVersion (p. 150)	Returns the Virtuozzo product version number.
	PrlLoginResponse_GetRunningTaskByIndex (p. 151)	Obtains a handle of type PHT_RUNNING_TASK (p. 217).
	PrlLoginResponse_GetRunningTaskCount (p. 152)	Returns the total number of PHT_RUNNING_TASK (p. 217) handles contained in the PHT_LOGIN_RESPONSE object.
	PrlLoginResponse_GetServerUuid (p. 152)	Returns the host machine UUID.
	PrlLoginResponse_GetSessionUuid (p. 153)	Returns session UUID string (used to restore a session).

PrlLoginResponse_GetHostOsVersion Function

Returns the host OS version.

Syntax

```
PRL_RESULT PrlLoginResponse_GetHostOsVersion(
    PRL_HANDLE hLoginResp,
    PRL_STR sHostOsVersion,
    PRL_UINT32_PTR pnHostOsVersionBufLength
);
```

File

PrlApiDisp.h

Parameters

hLoginResp

A handle of type PHT_LOGIN_RESPONSE (p. 149) containing the login response info.

sHostOsVersion

[out] A pointer to a buffer that receives the OS version (a UTF-8 encoded, null-terminated string).

pnHostOsVersionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLoginResponse_GetProductVersion Function

Returns the Virtuozzo product version number.

Syntax

```
PRL_RESULT PrlLoginResponse_GetProductVersion(
    PRL_HANDLE hLoginResp,
    PRL_STR sProductVersion,
    PRL_UINT32_PTR pnProductVersionBufLength
);
```

File

PrlApiDisp.h

Parameters

hLoginResp

A handle of type PHT_LOGIN_RESPONSE (p. 149) containing the login response info.

sProductVersion

[out] A pointer to a buffer that receives the product version (a UTF-8 encoded, null-terminated string).

pnProductVersionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLoginResponse_GetRunningTaskByIndex Function

Obtains a handle of type PHT_RUNNING_TASK (p. 217).

Syntax

```
PRL_RESULT PrlLoginResponse_GetRunningTaskByIndex(
    PRL_HANDLE hLoginResp,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phRunningTask
);
```

File

PrlApiDisp.h

Parameters

hLoginResp

A handle of type PHT_LOGIN_RESPONSE (p. 149) containing the login response info.

nIndex

An index of the task to obtain the handle for (starting at 0). To determine the total number of the available tasks, use the PrlLoginResponse_GetRunningTaskCount (p. 152) function.

phRunningTask

[out] A pointer to a variable that receives the handle of type PHT_RUNNING_TASK (p. 217) containing the task information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If a previous session ID was passed to the corresponding login operation, the PHT_LOGIN_RESPONSE (p. 149) object will contain a list of the tasks that were initiated during the previous session and are still running (if any). A task ID (that can be retrieved using this function) can be passed to the PrlSrv_AttachToLostTask (p. 224) function to get a handle to the task, thus gaining control to it.

PrlLoginResponse_GetRunningTaskCount Function

Returns the total number of PHT_RUNNING_TASK (p. 217) handles contained in the PHT_LOGIN_RESPONSE (p. 149) object.

Syntax

```
PRL_RESULT PrlLoginResponse_GetRunningTaskCount(
    PRL_HANDLE hLoginResp,
    PRL_UINT32_PTR pnCount
);
```

File

PrlApiDisp.h

Parameters

hLoginResp

A handle of type PHT_LOGIN_RESPONSE (p. 149) containing the login response info.

pnCount

[out] A pointer to a variable that receives the size of the array.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

See the PrlLoginResponse_GetRunningTaskByIndex (p. 151) function for more information.

PrlLoginResponse_GetServerUuid Function

Returns the host machine UUID.

Syntax

```
PRL_RESULT PrlLoginResponse_GetServerUuid(
    PRL_HANDLE hLoginResp,
    PRL_STR sServerUuid,
    PRL_UINT32_PTR pnServerUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hLoginResp

A handle of type PHT_LOGIN_RESPONSE (p. 149) containing the login response info.

sServerUuid

[out] A pointer to a buffer that receives the Virtuozzo Service UUID (a UTF-8 encoded, null-terminated string).

pnServerUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlLoginResponse_GetSessionUuid Function

Returns session UUID string (used to restore a session).

Syntax

```
PRL_RESULT PrlLoginResponse_GetSessionUuid(
    PRL_HANDLE hLoginResp,
    PRL_STR sSessionUuid,
    PRL_UINT32_PTR pnSessionUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hLoginResp

A handle of type PHT_LOGIN_RESPONSE (p. 149) containing the login response info.

sSessionUuid

[out] A pointer to a buffer that receives the session UUID (a UTF-8 encoded, null-terminated string).

pnSessionUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_NET_SERVICE_STATUS

The PHT_NET_SERVICE_STATUS handle provides methods for retrieving the Virtuozzo Service network status information. A

handle to it is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetNetServiceStatus (p. 249) function.

Functions

	Name	Description
	PrlNetSvc_GetStatus (p. 154)	Returns the Virtuozzo Service network status information.

PrlNetSvc_GetStatus Function

Returns the Virtuozzo Service network status information.

Syntax

```
PRL_RESULT PrlNetSvc_GetStatus(
    PRL_HANDLE handle,
    PRL_SERVICE_STATUS_ENUM_PTR pnNetStatus
);
```

File

PrlApiNet.h

Parameters

handle

A handle of type PHT_NET_SERVICE_STATUS (p. 153).

pnNetStatus

[out] A pointer to a variable that receives the status information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_NETWORK_CLASS

The PHT_NETWORK_CLASS handle is used to configure network accounting. To use the handle, first create it using the PrlNetworkClass_Create (p. 155) function and then use the handle function to manage network classes configuration.

Functions

	Name	Description
	PrlNetworkClass_Create (p. 155)	Creates a new handle of type PHT_NETWORK_CLASS.
	PrlNetworkClass_GetClassId (p. 155)	Returns the network class id.
	PrlNetworkClass_GetNetworkList (p. 156)	Returns the list of networks for specified network class entry.
	PrlNetworkClass_SetClassId (p. 156)	Sets the network class id.

	PrlNetworkClass_SetNetworkList (p. 157)	Sets the list of networks for specified network class entry.

PrlNetworkClass_Create Function

Creates a new handle of type PHT_NETWORK_CLASS (p. 154).

Syntax

```
PRL_RESULT PrlNetworkClass_Create(
    PRL_UINT32 nClassId,
    PRL_HANDLE_PTR phNetworkClass
);
```

File

PrlApiNet.h

Parameters

nClassId

Network class id.

phNetworkClass

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to configure network accounting configuration If you would like to manage network classes configuration, use this function to create the handle first and then use the handle functions to configure the Network class.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlNetworkClass_GetClassId Function

Returns the network class id.

Syntax

```
PRL_RESULT PrlNetworkClass_GetClassId(
    PRL_HANDLE hNetworkClass,
    PRL_UINT32_PTR pnClassId
);
```

File

PrlApiNet.h

Parameters

hNetworkClass

A handle of type PHT_NETWORK_CLASS (p. 154) identifying the network class entry.

pnClassId

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkClass_GetNetworkList Function

Returns the list of networks for specified network class entry.

Syntax

```
PRL_RESULT PrlNetworkClass_GetNetworkList(
    PRL_HANDLE hNetworkClass,
    PRL_HANDLE_PTR phNetworkList
);
```

File

PrlApiNet.h

Parameters

hNetworkClass

A handle of type PHT_NETWORK_CLASS (p. 154) identifying the network class entry.

phNetworkList

[out] A pointer to a handle of type PHT_STRINGS_LIST (p. 357) that receives the list of networks

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkClass_SetClassId Function

Sets the network class id.

Syntax

```
PRL_RESULT PrlNetworkClass_SetClassId(
    PRL_HANDLE hNetworkClass,
    PRL_UINT32 nClassId
```

```
);
```

File

PrlApiNet.h

Parameters

hNetworkClass

A handle of type PHT_NETWORK_CLASS (p. 154) identifying the network class entry.

nClassId

Class id

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkClass_SetNetworkList Function

Sets the list of networks for specified network class entry.

Syntax

```
PRL_RESULT PrlNetworkClass_SetNetworkList(
    PRL_HANDLE hNetworkClass,
    PRL_HANDLE hNetworkList
);
```

File

PrlApiNet.h

Parameters

hNetworkClass

A handle of type PHT_NETWORK_CLASS (p. 154) identifying the network class entry.

hNetworkList

A handle of type PHT_STRINGS_LIST (p. 357)

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_NETWORK_RATE

The PHT_NETWORK_RATE handle is used to configure the virtual machine output bandwidth. The handle is obtained using the

PrlNetworkRate_Create (p. 158) function.

Functions

	Name	Description
	PrlNetworkRate_Create (p. 158)	Creates a new handle of type PHT_NETWORK_RATE.
	PrlNetworkRate_GetClassId (p. 158)	Returns the network rate entry class id.
	PrlNetworkRate_GetRate (p. 159)	Returns the network rate entry Rate value.

PrlNetworkRate_Create Function

Creates a new handle of type PHT_NETWORK_RATE (p. 157).

Syntax

```
PRL_RESULT PrlNetworkRate_Create(
    PRL_UINT32 nClassId,
    PRL_UINT32 nRate,
    PRL_HANDLE_PTR phNetworkRate
);
```

File

PrlApiNet.h

Parameters

nClassId

Network class id

nRate

Specifies bound rate for network class

phNetwork

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to configure Vm output bandwidth.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlNetworkRate_GetClassId Function

Returns the network rate entry class id.

Syntax

```
PRL_RESULT PrlNetworkRate_GetClassId(
    PRL_HANDLE hNetworkRate,
    PRL_UINT32_PTR pnClassId
);
```

File

PrlApiNet.h

Parameters

hNetworkRate

A handle of type PHT_NETWORK_RATE (p. 157) identifying the network rate entry.

pnClassId

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkRate_GetRate Function

Returns the network rate entry Rate value.

Syntax

```
PRL_RESULT PrlNetworkRate_GetRate(
    PRL_HANDLE hNetworkRate,
    PRL_UINT32_PTR pnRate
);
```

File

PrlApiNet.h

Parameters

hNetworkRate

A handle of type PHT_NETWORK_RATE (p. 157) identifying the network rate entry.

pnRate

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_NETWORK_SHAPING

The PHT_NETWORK_SHAPING handle is used to configure network shaping. The handle is obtained using the PrlNetworkShapingEntry_Create (p. 160) function.

Functions

	Name	Description
	PrlNetworkShapingEntry_Create (p. 160)	Creates a new handle of type PHT_NETWORK_SHAPING.
	PrlNetworkShapingEntry_GetClassId (p. 161)	Returns the network class id.
	PrlNetworkShapingEntry_GetDevice (p. 161)	Returns the network device for specified shaping entry.
	PrlNetworkShapingEntry_GetRate (p. 162)	Returns the default rate for specified shaping entry.
	PrlNetworkShapingEntry_GetRateMPU (p. 162)	Returns MPU size for packet rate limitation.
	PrlNetworkShapingEntry_GetTotalRate (p. 163)	Returns the total rate for specified shaping entry.
	PrlNetworkShapingEntry_SetClassId (p. 163)	Sets the network class id for specified shaping entry.
	PrlNetworkShapingEntry_SetDevice (p. 164)	Sets the specified network device.
	PrlNetworkShapingEntry_SetRate (p. 164)	Sets the default network guaranteed rate.
	PrlNetworkShapingEntry_SetRateMPU (p. 165)	Allows to enable and disable packet rate limiting by setting MPU.
	PrlNetworkShapingEntry_SetTotalRate (p. 165)	Sets the total rate for specified shaping entry.

PrlNetworkShapingEntry_Create Function

Creates a new handle of type PHT_NETWORK_SHAPING (p. 160).

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_Create(
    PRL_UINT32 nClassId,
    PRL_UINT32 nTotalRate,
    PRL_HANDLE_PTR phNetworkShapingEntry
);
```

File

PrlApiNet.h

Parameters

nClassId

Network class id

nTotalRate

Specifies bound rate for network class phNetworkShapingEntry:[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to configure network shaping configuration.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlNetworkShapingEntry_GetClassId Function

Returns the network class id.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_GetClassId(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_UINT32_PTR pnClassId
);
```

File

PrlApiNet.h

Parameters

pnClassId

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_GetDevice Function

Returns the network device for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_GetDevice(
    PRL_HANDLE hNetworkShapingEntry,
    PRL_STR sDev,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiNet.h

Parameters

sDev

[out] A pointer to a buffer that receives the result.

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_GetRate Function

Returns the default rate for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_GetRate(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_UINT32_PTR pnRate
);
```

File

PrlApiNet.h

Parameters

pnRate

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_GetRateMPU Function

Returns MPU size for packet rate limitation.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_GetRateMPU(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_INT32_PTR pnRateMPU
);
```

File

PrlApiNet.h

Parameters

pNetworkShapingEntry

A handle of type PHT_NETWORK_SHAPING (p. 160) identifying the network shaping entry.

pnRateMPU

[out] A pointer to a variable that receives the return value. May be a number lower than MPU or one of the special values: NRM_DISABLED or NRM_ENABLED (for default value).

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_GetTotalRate Function

Returns the total rate for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_GetTotalRate(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_UINT32_PTR pnTotalRate
);
```

File

PrlApiNet.h

Parameters

pnTotalRate

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_SetClassId Function

Sets the network class id for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_SetClassId(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_UINT32 nClassId
);
```

File

PrlApiNet.h

Parameters

nClassId

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_SetDevice Function

Sets the specified network device.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_SetDevice(
    PRL_HANDLE hNetworkShapingEntry,
    PRL_CONST_STR sDev
);
```

File

PrlApiNet.h

Parameters

hNetworkShapingEntry

A handle of type PHT_NETWORK_SHAPING (p. 160) identifying the shaping entry.

sDev

The network device.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_SetRate Function

Sets the default network guaranteed rate.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_SetRate(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_UINT32 nRate
);
```

File

PrlApiNet.h

Parameters

hNetworkShapingEntry

A handle of type PHT_NERWORK_SHAPING identifying the shaping entry.

sRate

The network rate in Kbits.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_SetRateMPU Function

Allows to enable and disable packet rate limiting by setting MPU.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_SetRateMPU(
    PRL_HANDLE pNetworkShapingEntry,
    PRL_INT32 nRateMPU
);
```

File

PrlApiNet.h

Parameters

nRateMPU

An integer value indicating packet MPU. Packets with size lower than MPU will be threated as those of size MPU. May be a number lower than MTU or one of the special values: NRM_DISABLED or NRM_ENABLED (for default value).

hNetworkShapingEntry

A handle of type PHT_NERWORK_SHAPING identifying the shaping entry.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingEntry_SetTotalRate Function

Sets the total rate for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingEntry_SetTotalRate(
```

```
    PRL_HANDLE pNetworkShapingEntry,
    PRL_UINT32 nTotalRate
);
```

File

PrlApiNet.h

Parameters

nTotalRate

The network total rate in Kbits.

hNetworkShapingEntry

A handle of type PHT_NETWORK_SHAPING (p. 160) identifying the shaping entry.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_NETWORK_SHAPING_BANDWIDTH

The PHT_NETWORK_SHAPING_BANDWIDTH handle is used to create network shaping configuration.

Functions

	Name	Description
	PrlNetworkShapingBandwidthEntry_Create (p. 166)	Creates a new handle of type PHT_NETWORK_SHAPING_BANDWIDTH.
	PrlNetworkShapingBandwidthEntry_GetBandwidth (p. 167)	Returns the network device bandwidth for specified shaping entry.
	PrlNetworkShapingBandwidthEntry_GetDevice (p. 168)	Returns the network device name for specified shaping entry.
	PrlNetworkShapingBandwidthEntry_SetBandwidth (p. 168)	Sets the network device bandwidth for specified shaping entry.
	PrlNetworkShapingBandwidthEntry_SetDevice (p. 169)	Sets the network device name for specified shaping entry.

PrlNetworkShapingBandwidthEntry_Create Function

Creates a new handle of type PHT_NETWORK_SHAPING_BANDWIDTH (p. 166).

Syntax

```
PRL_RESULT PrlNetworkShapingBandwidthEntry_Create(
    PRL_CONST_STR sDev,
    PRL_UINT32 nBandwidth,
    PRL_HANDLE_PTR phDeviceEntry
);
```

File

PrlApiNet.h

Parameters

sDev

network device name.

nBandwidth

Network device bandwidth phDeviceEntryN:[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to configure network shaping configuration.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlNetworkShapingBandwidthEntry_GetBandwidth Function

Returns the network device bandwidth for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingBandwidthEntry_GetBandwidth(
    PRL_HANDLE hDeviceEntry,
    PRL_UINT32_PTR pnBandwidth
);
```

File

PrlApiNet.h

Parameters

hDeviceEntry

A handle of type PHT_NETWORK_SHAPING_BANDWIDTH (p. 166) identifying the network shaping entry.

pnBandwidth

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingBandwidthEntry_GetDevice Function

Returns the network device name for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingBandwidthEntry_GetDevice(
    PRL_HANDLE hDeviceEntry,
    PRL_STR sDev,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiNet.h

Parameters

hDeviceEntry

A handle of type PHT_NETWORK_SHAPING_BANDWIDTH (p. 166) identifying the network shaping entry.

sDev

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingBandwidthEntry_SetBandwidth Function

Sets the network device bandwidth for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingBandwidthEntry_SetBandwidth(
    PRL_HANDLE hDeviceEntry,
    PRL_UINT32 nBandwidth
);
```

File

PrlApiNet.h

Parameters

hDeviceEntry

A handle of type PHT_NETWORK_SHAPING_BANDWIDTH (p. 166) identifying the network shaping entry.

nBandwidth

Network device bandwidth

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingBandwidthEntry_SetDevice Function

Sets the network device name for specified shaping entry.

Syntax

```
PRL_RESULT PrlNetworkShapingBandwidthEntry_SetDevice(
    PRL_HANDLE hDeviceEntry,
    PRL_CONST_STR sDev
);
```

File

PrlApiNet.h

Parameters

hDeviceEntry

A handle of type PHT_NETWORK_SHAPING_BANDWIDTH (p. 166) identifying the network shaping entry.

sDev

network device name

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_NETWORK_SHAPING_CONFIG

The PHT_NETWORK_SHAPING_CONFIG handle is used to configure network shaping.

Functions

	Name	Description
	PrlNetworkShapingConfig_GetNetworkDeviceBandwidthList (p. 170)	Obtains a list of network shaping bandwidth entries represents the network shaping configuration.
	PrlNetworkShapingConfig_GetNetworkShapingList (p. 170)	Obtains a list of network shaping entries represents the network shaping configuration.
	PrlNetworkShapingConfig_IsEnabled (p. 171)	Determines whether the network shaping is enabled or disabled.
	PrlNetworkShapingConfig_SetEnabled (p. 171)	Allows to enable or disable the network shaping.
	PrlNetworkShapingConfig_SetNetworkDeviceBandwidthList (p. 172)	Update the list of network device bandwidth represents the network shaping configuration.
	PrlNetworkShapingConfig_SetNetworkShapingList (p. 172)	Update the list of network shaping entries represents the network shaping configuration.

PrlNetworkShapingConfig_GetNetworkDeviceBandwidthList Function

Obtains a list of network shaping bandwidth entries represents the network shaping configuration.

Syntax

```
PRL_RESULT PrlNetworkShapingConfig_GetNetworkDeviceBandwidthList(
    PRL_HANDLE hConfig,
    PRL_HANDLE_PTR phList
);
```

File

PrlApiNet.h

Parameters

hConfig

A valid handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169) identifying the adapter.

hList

A pointer to a handle of type PHT_HANDLE_LIST which contains the list of handles of type PHT_NETWORK_SHAPING (p. 160)

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingConfig_GetNetworkShapingList Function

Obtains a list of network shaping entries represents the network shaping configuration.

Syntax

```
PRL_RESULT PrlNetworkShapingConfig_GetNetworkShapingList(
    PRL_HANDLE hConfig,
    PRL_HANDLE_PTR phList
);
```

File

PrlApiNet.h

Parameters

hConfig

A valid handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169) identifying the adapter.

hList

A pointer to a handle of type PHT_HANDLE_LIST which contains the list of handles of type PHT_NETWORK_SHAPING (p. 160)

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingConfig_IsEnabled Function

Determines whether the network shaping is enabled or disabled.

Syntax

```
PRL_RESULT PrlNetworkShapingConfig_IsEnabled(
    PRL_HANDLE hConfig,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiNet.h

Parameters

hConfig

A valid handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169) identifying the adapter.

pbEnabled

[out] A pointer to a variable that receives the return value. PRL_TRUE (p. 1652) indicates that the network shaping is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingConfig_SetEnabled Function

Allows to enable or disable the network shaping.

Syntax

```
PRL_RESULT PrlNetworkShapingConfig_SetEnabled(
    PRL_HANDLE hConfig,
    PRL_BOOL bEnabled
);
```

File

PrlApiNet.h

Parameters

hConfig

A handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169).

bEnabled

A boolean value indicating the type of action to perform. PRL_TRUE (p. 1652) will enable the network shaping. PRL_FALSE (p. 1626) will disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingConfig_SetNetworkDeviceBandwidthList Function

Update the list of network device bandwidth represents the network shaping configuration.

Syntax

```
PRL_RESULT PrlNetworkShapingConfig_SetNetworkDeviceBandwidthList(
    PRL_HANDLE hConfig,
    PRL_HANDLE hList
);
```

File

PrlApiNet.h

Parameters

hConfig

A valid handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169) identifying the adapter.

hList

A handle of type PHT_HANDLE_LIST contains the list of handles of type PHT_NETWORK_SHAPING (p. 160)

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlNetworkShapingConfig_SetNetworkShapingList Function

Update the list of network shaping entries represents the network shaping configuration.

Syntax

```
PRL_RESULT PrlNetworkShapingConfig_SetNetworkShapingList(
    PRL_HANDLE hConfig,
    PRL_HANDLE hList
);
```

File

PrlApiNet.h

Parameters

hConfig

A valid handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169) identifying the adapter.

hList

A handle of type PHT_HANDLE_LIST contains the list of handles of type PHT_NETWORK_SHAPING (p. 160)

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_OFFLINE_SERVICE

The PHT_OFFLINE_SERVICE handle is used to configure offline services. When creating a new offline service configuration, use the PrlOffmgmtService_Create (p. 173) to create the handle. When managing existing offline services configurations, use the PrlSrv_GetOfflineServicesList (p. 252) function to obtain the list of handles.

Functions

	Name	Description
	PrlOffmgmtService_Create (p. 173)	Creates a new handle of type PHT_OFFLINE_SRVICE.
	PrlOffmgmtService_GetName (p. 174)	Returns the name of the specified offline service.
	PrlOffmgmtService_GetPort (p. 175)	Returns the port of the specified offline service.
	PrlOffmgmtService_IsUsedByDefault (p. 175)	Returns the type of the flag of the specified offline service used by default.
	PrlOffmgmtService_SetName (p. 176)	Sets the specified offline service name.
	PrlOffmgmtService_SetPort (p. 176)	Sets the specified offline service port.
	PrlOffmgmtService_SetUsedByDefault (p. 177)	Sets the default flag type for the specified offline service.

PrlOffmgmtService_Create Function

Creates a new handle of type PHT_OFFLINE_SRVICE.

Syntax

```
PRL_RESULT PrlOffmgmtService_Create(
    PRL_HANDLE_PTR phOffmgmtService
);
```

File

PrlApiNet.h

Parameters

phOffmgmtService

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to configure offline services. If you would like to manage offline service configuration, use this function to create the handle first and then use the handle functions to configure the offline services.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlOffmgmtService_GetName Function

Returns the name of the specified offline service.

Syntax

```
PRL_RESULT PrlOffmgmtService_GetName(
    PRL_HANDLE hOffmgmtService,
    PRL_STR sName,
    PRL_UINT32_PTR pnNameBufLength
);
```

File

PrlApiNet.h

Parameters

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying the offline service.

pnNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

sname

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOffmgmtService_GetPort Function

Returns the port of the specified offline service.

Syntax

```
PRL_RESULT PrlOffmgmtService_GetPort(
    PRL_HANDLE hOffmgmtService,
    PRL_UINT32_PTR pnPort
);
```

File

PrlApiNet.h

Parameters

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying the offline service.

sname

[out] A pointer to a buffer that receives the result

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOffmgmtService_IsUsedByDefault Function

Returns the type of the flag of the specified offline service used by default.

Syntax

```
PRL_RESULT PrlOffmgmtService_IsUsedByDefault(
    PRL_HANDLE hOffmgmtService,
    PRL_BOOL_PTR pbDefault
);
```

File

PrlApiNet.h

Parameters

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying the offline service.

sname

[out] A pointer to a buffer that receives the result

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOffmgmtService_SetName Function

Sets the specified offline service name.

Syntax

```
PRL_RESULT PrlOffmgmtService_SetName(
    PRL_HANDLE hOffmgmtService,
    PRL_CONST_STR sName
);
```

File

PrlApiNet.h

Parameters

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying the offline service.

sName

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOffmgmtService_SetPort Function

Sets the specified offline service port.

Syntax

```
PRL_RESULT PrlOffmgmtService_SetPort(
    PRL_HANDLE hOffmgmtService,
    PRL_UINT32 nPort
);
```

File

PrlApiNet.h

Parameters

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying the offline service.

nPort

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOffmgmtService_SetUsedByDefault Function

Sets the default flag type for the specified offline service.

Syntax

```
PRL_RESULT PrlOffmgmtService_SetUsedByDefault(
    PRL_HANDLE hOffmgmtService,
    PRL_BOOL bDefault
);
```

File

PrlApiNet.h

Parameters

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying the offline service.

bDefault

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_OPAQUE_TYPE_LIST

The PHT_OPAQUE_TYPE_LIST handle is a container for storing type lists.

Functions

	Name	Description
	PrlOpTypeList_AddItem (p. 178)	Adds a new item to the container.
	PrlOpTypeList_GetItem (p. 178)	Extracts the specified item from the container.
	PrlOpTypeList_GetItemsCount (p. 179)	Determines the number of items in the container.
	PrlOpTypeList_GetTypeSize (p. 179)	Extracts the size of type of items storing in container.
	PrlOpTypeList_RemoveItem (p. 180)	Removes the specified item from the container.

PrlOpTypeList_AddItem Function

Adds a new item to the container.

Syntax

```
PRL_RESULT PrlOpTypeList_AddItem(
    PRL_HANDLE hOpTypeList,
    PRL_CONST_VOID_PTR pnItem
);
```

File

PrlApiCore.h

Parameters

hOpTypeList

A handle of type PHT_OPAQUE_TYPE_LIST (p. 177) identifying the opaque type container.

pnItem

A pointer to the item to add to the container. Note that value passing through pointer in order to prevent loosing of precision (for 64 bit types for example).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to value variable was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOpTypeList_GetItem Function

Extracts the specified item from the container.

Syntax

```
PRL_RESULT PrlOpTypeList_GetItem(
    PRL_HANDLE hOpTypeList,
    PRL_UINT32 nItemIndex,
    PRL_VOID_PTR pnItem
);
```

File

PrlApiCore.h

Parameters

hOpTypeList

A handle of type PHT_OPAQUE_TYPE_LIST (p. 177) identifying the opaque type container.

nItemIndex

A numeric index identifying the item (beginning with 0). To determine the number of items in the container, use the PrlOpTypeList_GetItemsCount (p. 179) function.

pnItem

[in] The pointer to the buffer to extract item value.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer or out of range index was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOpTypeList_GetItemsCount Function

Determines the number of items in the container.

Syntax

```
PRL_RESULT PrlOpTypeList_GetItemsCount(
    PRL_HANDLE hOpTypeList,
    PRL_UINT32_PTR pnItemsCount
);
```

File

PrlApiCore.h

Parameters

hOpTypeList

A handle of type PHT_OPAQUE_TYPE_LIST (p. 177) identifying the opaque type container.

pnItemsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOpTypeList_GetTypeSize Function

Extracts the size of type of items storing in container.

Syntax

```
PRL_RESULT PrlOpTypeList_GetTypeSize(
    PRL_HANDLE hOpTypeList,
    PRL_SIZE_PTR pnTypeSize
);
```

File

PrlApiCore.h

Parameters

hOpTypeList

A handle of type PHT_OPAQUE_TYPE_LIST (p. 177) identifying the opaque type container.

pnTypeSize

[out] The pointer to the buffer to extract type size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlOpTypeList_RemoveItem Function

Removes the specified item from the container.

Syntax

```
PRL_RESULT PrlOpTypeList_RemoveItem(
    PRL_HANDLE hOpTypeList,
    PRL_UINT32 nItemIndex
);
```

File

PrlApiCore.h

Parameters

hOpTypeList

A handle of type PHT_OPAQUE_TYPE_LIST (p. 177) identifying the opaque type container.

nItemIndex

A numeric index identifying the item. Index is a sequential number assigned to every item in the container. To determine the index of an item, call the PrlOpTypeList_GetItem (p. 178) in a loop (from 0 to PrlOpTypeList_GetItemsCount (p. 179)()). The iteration number is the index of the corresponding item.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or out of range index was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_PLUGIN_INFO

The PHT_PLUGIN_INFO handle is a container that contains information about a third-party plug-in installed in Virtuozzo Service. To obtain the list of installed plug-ins use the PrlSrv_GetPluginsList (p. 254) function.

Functions

	Name	Description
	PrlPluginInfo_GetCopyright (p. 181)	Returns copyright of the plugin.
	PrlPluginInfo_GetId (p. 181)	Returns identificator of the plugin in GUID format.
	PrlPluginInfo_GetLongDescription (p. 182)	Returns long description about the plugin.

	PrlPluginInfo_GetShortDescription (p. 183)	Returns short description about the plugin.
	PrlPluginInfo_GetVendor (p. 183)	Returns the vendor of the plugin.
	PrlPluginInfo_GetVersion (p. 184)	Returns the version of the plugin.

PrlPluginInfo_GetCopyright Function

Returns copyright of the plugin.

Syntax

```
PRL_RESULT PrlPluginInfo_GetCopyright(
    PRL_HANDLE hPluginInfo,
    PRL_STR sCopyright,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hPluginInfo

A handle of type PHT_PLUGIN_INFO (p. 180) identifying the plugin info.

sCopyright

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no information about this field.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPluginInfo_GetId Function

Returns identificator of the plugin in GUID format.

Syntax

```
PRL_RESULT PrlPluginInfo_GetId(
    PRL_HANDLE hPluginInfo,
    PRL_STR sId,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hPluginInfo

A handle of type PHT_PLUGIN_INFO (p. 180) identifying the plugin info.

sId

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no information about this field.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPluginInfo_GetLongDescription Function

Returns long description about the plugin.

Syntax

```
PRL_RESULT PrlPluginInfo_GetLongDescription(
    PRL_HANDLE hPluginInfo,
    PRL_STR sDescr,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hPluginInfo

A handle of type PHT_PLUGIN_INFO (p. 180) identifying the plugin info.

sDescr

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no information about this field.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPluginInfo_GetShortDescription Function

Returns short description about the plugin.

Syntax

```
PRL_RESULT PrlPluginInfo_GetShortDescription(
    PRL_HANDLE hPluginInfo,
    PRL_STR sDescr,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hPluginInfo

A handle of type PHT_PLUGIN_INFO (p. 180) identifying the plugin info.

sDescr

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no information about this field.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPluginInfo_GetVendor Function

Returns the vendor of the plugin.

Syntax

```
PRL_RESULT PrlPluginInfo_GetVendor(
    PRL_HANDLE hPluginInfo,
```

```
    PRL_STR sVendor,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hPluginInfo

A handle of type PHT_PLUGIN_INFO (p. 180) identifying the plugin info.

sVendor

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no information about this field.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPluginInfo_GetVersion Function

Returns the version of the plugin.

Syntax

```
PRL_RESULT PrlPluginInfo_GetVersion(
    PRL_HANDLE hPluginInfo,
    PRL_STR sVersion,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiDisp.h

Parameters

hPluginInfo

A handle of type PHT_PLUGIN_INFO (p. 180) identifying the plugin info.

sVersion

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string). Format version is "Major.Minor.Build".

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no information about this field.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_PORT_FORWARDING

The PHT_PORT_FORWARDING handle provides access to the Port Forwarding functionality. Using this functionality, you can redirect all incoming data from a specific TCP or UDP port in the host machine to the specified port in a virtual machine.

Description

Port forwarding is used when you have a virtual machine communicating with the outside world using shared networking and would like other computers to be able to access certain services running in it (e.g. HTTP server, FTP server, or any other service listening for client requests on a particular port). In order to do so, you have to tell the host machine to forward all data that it receives on a certain port to the appropriate port in the virtual machine. This is accomplished by creating a port forwarding entry using the PHT_PORT_FORWARDING handle. A port forwarding entry has the following properties:

- The source port number in the host computer. All data that the host computer receives on this port will be forwarded to the virtual machine.
- The destination virtual machine identified by its IP address. The address can be determined using the standard tools of the guest operating system.
- The destination port number in the virtual machine. All data will be forwarded from the source port on the host machine to this port.

To obtain a list of the handles containing information about existing port forwarding entries, use the PrlVirtNet_GetPortForwardList (p. 520) function. To create a new port forwarding entry, use PrlPortFwd_Create (p. 186) to create a new handle. Use the functions describes in this section to define or modify the port forwarding entry properties and then use the PrlVirtNet_SetPortForwardList (p. 535) function to save it.

Functions

	Name	Description
	PrlPortFwd_Create (p. 186)	Creates a new handle of type PHT_PORT_FORWARDING.
	PrlPortFwd_GetIncomingPort (p. 186)	Returns the incoming port.
	PrlPortFwd_GetRedirectIPAddress (p. 187)	Returns the redirect IP address of the specified port forward entry.
	PrlPortFwd_GetRedirectPort (p. 187)	Returns the redirect port.
	PrlPortFwd_GetRedirectVm (p. 188)	Returns the redirect virtual machine identifier of the specified port forward entry.
	PrlPortFwd_GetRuleName (p. 189)	Returns the rule name of the specified port forward entry.
	PrlPortFwd_SetIncomingPort (p. 189)	Sets the specified incoming port.
	PrlPortFwd_SetRedirectIPAddress (p. 190)	Sets the specified port forwardi entry redirect IP address.

	PrlPortFwd_SetRedirectPort (p. 190)	Sets the specified redirect port.
	PrlPortFwd_SetRedirectVm (p. 191)	Sets the specified port forward entry virtual machine identifier.
	PrlPortFwd_SetRuleName (p. 191)	Sets the specified port forward entry rule name.

PrlPortFwd_Create Function

Creates a new handle of type PHT_PORT_FORWARDING (p. 185).

Syntax

```
PRL_RESULT PrlPortFwd_Create(
    PRL_HANDLE_PTR phPortFwd
);
```

File

PrlApiNet.h

Parameters

phPortFwd

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a port forward handle object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It is the responsibility of the user to free the handle when it is no longer needed.

PrlPortFwd_GetIncomingPort Function

Returns the incoming port.

Syntax

```
PRL_RESULT PrlPortFwd_GetIncomingPort(
    PRL_HANDLE hPortFwd,
    PRL_UINT16_PTR pnIncomingPort
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

pnIncomingPort

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_GetRedirectIPAddress Function

Returns the redirect IP address of the specified port forward entry.

Syntax

```
PRL_RESULT PrlPortFwd_GetRedirectIPAddress(
    PRL_HANDLE hPortFwd,
    PRL_STR sRedirectIPAddress,
    PRL_UINT32_PTR pnRedirectIPAddressBufLength
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

sRedirectIPAddress

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnRedirectIPAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_GetRedirectPort Function

Returns the redirect port.

Syntax

```
PRL_RESULT PrlPortFwd_GetRedirectPort(
    PRL_HANDLE hPortFwd,
    PRL_UINT16_PTR pnRedirectPort
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

pnRedirectPort

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_GetRedirectVm Function

Returns the redirect virtual machine identifier of the specified port forward entry.

Syntax

```
PRL_RESULT PrlPortFwd_GetRedirectVm(
    PRL_HANDLE hPortFwd,
    PRL_STR sRedirectVm,
    PRL_UINT32_PTR pnRedirectVmBufLength
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

sRedirectVm

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnRedirectVmBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_GetRuleName Function

Returns the rule name of the specified port forward entry.

Syntax

```
PRL_RESULT PrlPortFwd_GetRuleName(
    PRL_HANDLE hPortFwd,
    PRL_STR sRuleName,
    PRL_UINT32_PTR pnRuleNameBufLength
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

sRuleName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnRuleNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_SetIncomingPort Function

Sets the specified incoming port.

Syntax

```
PRL_RESULT PrlPortFwd_SetIncomingPort(
    PRL_HANDLE hPortFwd,
    PRL_UINT16 nIncomingPort
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

nIncomingPort

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_SetRedirectIPAddress Function

Sets the specified port forwardi entry redirect IP address.

Syntax

```
PRL_RESULT PrlPortFwd_SetRedirectIPAddress(
    PRL_HANDLE hPortFwd,
    PRL_CONST_STR sRedirectIPAddress
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

sRedirectIPAddress

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_SetRedirectPort Function

Sets the specified redirect port.

Syntax

```
PRL_RESULT PrlPortFwd_SetRedirectPort(
    PRL_HANDLE hPortFwd,
    PRL_UINT16 nRedirectPort
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

nRedirectPort

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_SetRedirectVm Function

Sets the specified port forward entry virtual machine identifier.

Syntax

```
PRL_RESULT PrlPortFwd_SetRedirectVm(
    PRL_HANDLE hPortFwd,
    PRL_CONST_STR sRedirectVm
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

sRedirectVm

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlPortFwd_SetRuleName Function

Sets the specified port forward entry rule name.

Syntax

```
PRL_RESULT PrlPortFwd_SetRuleName(
    PRL_HANDLE hPortFwd,
    PRL_CONST_STR sRuleName
);
```

File

PrlApiNet.h

Parameters

hPortFwd

A handle of type PHT_PORT_FORWARDING (p. 185) identifying the port forwarding.

sRuleName

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_PROBLEM_REPORT

The PHT_PROBLEM_REPORT handle is used to obtain and manage Virtuozzo Problem Reports. To obtain the handle, use the PrlApi_CreateProblemReport (p. 729) function.

Functions

	Name	Description
	PrlReport_Assembly (p. 192)	Assembles the problem report.
	PrlReport_AsString (p. 193)	Returns XML based string representation of problem report.
	PrlReport_GetArchiveFileName (p. 194)	Returns problem report archive file name which can be used for further problem report data saving and processing.
	PrlReport_GetData (p. 194)	Returns the binary representation of problem report.
	PrlReport_GetDescription (p. 195)	Obtains the problem report description.
	PrlReport_GetReason (p. 196)	Obtains the value of the property, identifying the reason for this project report.
	PrlReport_GetScheme (p. 196)	Determines the scheme used by the problem report object.
	PrlReport_GetType (p. 197)	Determines the type of the problem report object.
	PrlReport_GetUserEmail (p. 197)	Obtains the e-mail address of the user on the problem report.
	PrlReport_GetUserName (p. 198)	Obtains the name of the user on the problem report.
	PrlReport_Send (p. 198)	The PrlReport_Send function is provided for convenience.
	PrlReport_SetDescription (p. 200)	Sets the specified problem report description.
	PrlReport_SetReason (p. 200)	Sets the value of the object property, identifying the reason for this problem report.
	PrlReport_SetType (p. 201)	Sets the problem report object type.
	PrlReport_SetUserEmail (p. 201)	Sets the e-mail address of the user on the problem report.
	PrlReport_SetUserName (p. 202)	Sets the name of the user on the problem report.

PrlReport_Assembly Function

Assembles the problem report.

Syntax

```
PRL_HANDLE PrlReport_Assembly(
    PRL_HANDLE hProblemReport,
    PRL_UINT32 nFlags
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the container.

nFlags

[in] flags bitset (accepts values just from PRL_PROBLEM_REPORT_FLAGS (p. 1086))

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

PrlReport_AsString Function

Returns XML based string representation of problem report.

Syntax

```
PRL_RESULT PrlReport_AsString(
    PRL_HANDLE hProblemReport,
    PRL_STR sBuffer,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sBuffer

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetArchiveFileName Function

Returns problem report archive file name which can be used for further problem report data saving and processing.

Syntax

```
PRL_RESULT PrlReport_GetArchiveFileName(
    PRL_HANDLE hProblemReport,
    PRL_STR sArchiveFileName,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sArchiveFileName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or old XML based scheme problem report object handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Please note that this data makes sense just for packed problem report scheme.

PrlReport_GetData Function

Returns the binary representation of problem report.

Syntax

```
PRL_RESULT PrlReport_GetData(
    PRL_HANDLE hProblemReport,
    PRL_VOID_PTR pBuffer,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the container.

pBuffer

[out] A pointer to a buffer that receives the result.

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetDescription Function

Obtains the problem report description.

Syntax

```
PRL_RESULT PrlReport_GetDescription(
    PRL_HANDLE hProblemReport,
    PRL_STR sDescription,
    PRL_UINT32_PTR pnDescriptionBufLength
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sDescription

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDescriptionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetReason Function

Obtains the value of the property, identifying the reason for this project report.

Syntax

```
PRL_RESULT PrlReport_GetReason(
    PRL_HANDLE hProblemReport,
    PRL_PROBLEM_REPORT_REASON_PTR pnReportReason
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192).

pnReportReason

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or wrong handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetScheme Function

Determines the scheme used by the problem report object.

Syntax

```
PRL_RESULT PrlReport_GetScheme(
    PRL_HANDLE hProblemReport,
    PRL_PROBLEM_REPORT_SCHEME_PTR pnReportScheme
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] a handle of type PHT_PROBLEM_REPORT (p. 192).

pnReportScehme

[out] a pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or wrong handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetType Function

Determines the type of the problem report object.

Syntax

```
PRL_RESULT PrlReport_GetType(
    PRL_HANDLE hProblemReport,
    PRL_PROBLEM_REPORT_TYPE_PTR pnReportType
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192).

pnReportType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or wrong handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetUserEmail Function

Obtains the e-mail address of the user on the problem report.

Syntax

```
PRL_RESULT PrlReport_GetUserEmail(
    PRL_HANDLE hProblemReport,
    PRL_STR sUserEmail,
    PRL_UINT32_PTR pnUserEmailBufLength
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sUserEmail

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnUserEmailBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required

size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_GetUserName Function

Obtains the name of the user on the problem report.

Syntax

```
PRL_RESULT PrlReport_GetUserName(
    PRL_HANDLE hProblemReport,
    PRL_STR sUserName,
    PRL_UINT32_PTR pnUserNameBufLength
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sUserName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnUserNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_Send Function

The PrlReport_Send function is provided for convenience.

Syntax

```
PRL_HANDLE PrlReport_Send(
    PRL_HANDLE hProblemReport,
    PRL_BOOL bUseProxy,
    PRL_CONST_STR sProxyHost,
    PRL_UINT32 nProxyPort,
    PRL_CONST_STR sProxyUserLogin,
    PRL_CONST_STR sProxyUserPasswd,
    PRL_UINT32 nProblemSendTimeout,
    PRL_UINT32 nReserved,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR pUserData
);
```

File

PrlApiCore.h

Parameters

hProblemReport

A handle of type PHT_PROBLEM_REPORT (p. 192) containing report data.

bUseProxy

Specifies whether proxy should be used. The call can obtain and use the system proxy settings or you can specify your own settings. Note that custom values override system settings.

sProxyHost

[optional] If using proxy, this parameter allows to specify the proxy server host address/name (can be null or empty).

nProxyPort

[optional] If using proxy, this parameter allows to specify the proxy server port (to use the system setting, set this parameter to 0).

sProxyUserLogin

[optional] If using proxy, this parameter allows to specify user login if it??™s not set in the system settings (can be null or empty).

sProxyUserPasswd

[optional] If using proxy, this parameter allows to specify the user password if it??™s not specified in the system settings.

nProblemSendTimeout

Timeout in milliseconds. Can be set to 0 for unlimited timeout.

nReserved

[unused] Reserved parameter for future use.

handler

[optional] A pointer to a user callback function (event handler) that will be called every time a problem report sending progress is updated or when the job is completed.

pUserData

[optional] A pointer to a user data that will be passed to the optional callback function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It allows submitting a previously generated problem report to the Virtuozzo support server. The report could be generated using the PrlSrv_GetPackedProblemReport (p. 252) (PrlSrv_GetProblemReport (p. 254)) or PrlVm_GetProblemReport (p. 466) (PrlVm_GetPackedProblemReport (p. 465)) functions, or created and assembled using PrlApi_CreateProblemReport (p. 729)/PrlReport_Assembly (p. 192) functions.

PrlReport_SetDescription Function

Sets the specified problem report description.

Syntax

```
PRL_RESULT PrlReport_SetDescription(
    PRL_HANDLE hProblemReport,
    PRL_CONST_STR sNewDescription
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sNewDescription

[in] The value to set - must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_SetReason Function

Sets the value of the object property, identifying the reason for this problem report.

Syntax

```
PRL_RESULT PrlReport_SetReason(
    PRL_HANDLE hProblemReport,
    PRL_PROBLEM_REPORT_REASON nReportReason
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192).

nReportReason

[in] The problem report reason value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - wrong handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_SetType Function

Sets the problem report object type.

Syntax

```
PRL_RESULT PrlReport_SetType(
    PRL_HANDLE hProblemReport,
    PRL_PROBLEM_REPORT_TYPE nReportType
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192).

nReportType

[in] The problem report object type to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - wrong handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_SetUserEmail Function

Sets the e-mail address of the user on the problem report.

Syntax

```
PRL_RESULT PrlReport_SetUserEmail(
    PRL_HANDLE hProblemReport,
    PRL_CONST_STR sNewUserEmail
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sNewUserEmail

[in] The value to set - must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlReport_SetUserName Function

Sets the name of the user on the problem report.

Syntax

```
PRL_RESULT PrlReport_SetUserName(
    PRL_HANDLE hProblemReport,
    PRL_CONST_STR sNewUserName
);
```

File

PrlApiCore.h

Parameters

hProblemReport

[in] A handle of type PHT_PROBLEM_REPORT (p. 192) identifying the problem report object.

sNewUserName

[in] The value to set - must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_REMOTE_FILESYSTEM_ENTRY

The PHT_REMOTE_FILESYSTEM_ENTRY handle, together with the PHT_REMOTE_FILESYSTEM_INFO (p. 206) handle, is used to obtain information about the file system on the host machine.The PHT_REMOTE_FILESYSTEM_ENTRY handle is a container that contains information about a single file system entry. The handle is obtained by making the PrlFsInfo_GetChildEntry (p. 207) call.

Functions

	Name	Description
	PrlFsEntry_GetAbsolutePath (p. 203)	Returns the specified file system entry absolute path.

	PrlFsEntry_GetLastModifiedDate (p. 203)	Returns the date on which the specified file system entry was last modified.
	PrlFsEntry_GetPermissions (p. 204)	Returns the specified file system entry permissions (read, write, execute) for the current user.
	PrlFsEntry_GetRelativeName (p. 205)	Returns the file system entry relative name.
	PrlFsEntry_GetSize (p. 205)	Returns the file system entry size.
	PrlFsEntry_GetType (p. 206)	Returns the file system entry type (file, directory, drive).

PrlFsEntry_GetAbsolutePath Function

Returns the specified file system entry absolute path.

Syntax

```
PRL_RESULT PrlFsEntry_GetAbsolutePath(
    PRL_HANDLE hFsEntry,
    PRL_STR sAbsolutePath,
    PRL_UINT32_PTR pnAbsolutePathBufLength
);
```

File

PrlApiDisp.h

Parameters

hFsEntry

A handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) identifying the file system entry.

sAbsolutePath

[out] A pointer to a buffer that receives the path (a UTF-8 encoded, null-terminated string).

pnAbsolutePathBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsEntry_GetLastModifiedDate Function

Returns the date on which the specified file system entry was last modified.

Syntax

```
PRL_RESULT PrlFsEntry_GetLastModifiedDate(
    PRL_HANDLE hFsEntry,
```

```
    PRL_STR sFsEntryLastModifiedDate,
    PRL_UINT32_PTR pnFsEntryLastModifiedDateBufLength
);
```

File

PrlApiDisp.h

Parameters

hFsEntry

A handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) identifying the file system entry.

sFsEntryLastModifiedDate

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string). The date is specified as: yyyy-mm-dd hh:mi:ss.

pnFsEntryLastModifiedDateBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Returned time is a local time (i.e. the time value is returned already converted to local time zone).

PrlFsEntry_GetPermissions Function

Returns the specified file system entry permissions (read, write, execute) for the current user.

Syntax

```
PRL_RESULT PrlFsEntry_GetPermissions(
    PRL_HANDLE hFsEntry,
    PRL_UINT32_PTR pnPermissions
);
```

File

PrlApiDisp.h

Parameters

hFsEntry

A handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) identifying the file system entry.

pnPermissions

[out] A pointer to a variable that receives the permission information. The permissions are specified as bitmasks. For bitmask values see macros in the PrlFs.h file.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsEntry_GetRelativeName Function

Returns the file system entry relative name.

Syntax

```
PRL_RESULT PrlFsEntry_GetRelativeName(
    PRL_HANDLE hFsEntry,
    PRL_STR sRelativeName,
    PRL_UINT32_PTR pnRelativeNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hFsEntry

A handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) identifying the file system entry.

sRelativeName

[out] A pointer to a buffer that receives the name (a UTF-8 encoded, null-terminated string).

pnRelativeNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsEntry_GetSize Function

Returns the file system entry size.

Syntax

```
PRL_RESULT PrlFsEntry_GetSize(
    PRL_HANDLE hFsEntry,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiDisp.h

Parameters

hFsEntry

A handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) identifying the file system entry.

pnSize

[out] A pointer to a variable that receives the size (in bytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsEntry_GetType Function

Returns the file system entry type (file, directory, drive).

Syntax

```
PRL_RESULT PrlFsEntry_GetType(
    PRL_HANDLE hFsEntry,
    PRL_FILE_SYSTEM_ELEMENT_TYPE_PTR pnFsEntryType
);
```

File

PrlApiDisp.h

Parameters

hFsEntry

A handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) identifying the file system entry.

pnFsEntryType

[out] A pointer to a variable that receives the type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_REMOTE_FILESYSTEM_INFO

The PHT_REMOTE_FILESYSTEM_INFO handle is a container that contains information about a file system entry and its immediate child elements (e.g. directory/files). The handle is obtained from a PHT_JOB (p. 138) object returned by the PrlSrv_FsGetDirEntries (p. 239) or the PrlSrv_FsGetDiskList (p. 240) function.

Description

To browse the file system on the host, first obtain a handle of type PHT_REMOTE_FILESYSTEM_INFO using

PrlSrv_FsGetDirEntries (p. 239) or the PrlSrv_FsGetDiskList (p. 240) function. The object will contain the information about the specified file system entry and all its child entries. Use the functions provided by the PHT_REMOTE_FILESYSTEM_INFO object to obtain a handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) containing the detailed information about the parent entry or any of its child entries. Use the functions provided by the PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) to get the individual properties of the entry.

Functions

	Name	Description
	PrlFsInfo_GetChildEntriesCount (p. 207)	Determines the number of child entries for the specified remote file system entry.
	PrlFsInfo_GetChildEntry (p. 207)	Obtains a handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) containing a child entry information.
	PrlFsInfo_GetFsType (p. 208)	Determines the file system type of the specified file system entry.
	PrlFsInfo_GetParentEntry (p. 209)	Obtains a handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) to the parent file system entry.
	PrlFsInfo_GetType (p. 209)	Determines the basic type of the specified file system entry.

PrlFsInfo_GetChildEntriesCount Function

Determines the number of child entries for the specified remote file system entry.

Syntax

```
PRL_RESULT PrlFsInfo_GetChildEntriesCount(
    PRL_HANDLE hFsInfo,
    PRL_UINT32_PTR pnChildEntriesCount
);
```

File

PrlApiDisp.h

Parameters

hFsInfo

A handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206) identifying the remote file system entry.

pnChildEntriesCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsInfo_GetChildEntry Function

Obtains a handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) containing a child entry information.

Syntax

```
PRL_RESULT PrlFsInfo_GetChildEntry(
```

```
    PRL_HANDLE hFsInfo,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phChildEntry
);
```

File

PrlApiDisp.h

Parameters

hFsInfo

A handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206) identifying the parent file system entry.

nIndex

A child entry index. To get the total number of child entries, use the PrlFsInfo_GetChildEntriesCount (p. 207) function.

phChildEntry

[out] A pointer to a variable that receives the child entry handle. The handle type is PHT_REMOTE_FILESYSTEM_ENTRY (p. 202).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsInfo_GetFsType Function

Determines the file system type of the specified file system entry.

Syntax

```
PRL_RESULT PrlFsInfo_GetFsType(
    PRL_HANDLE hFsInfo,
    PRL_FILE_SYSTEM_FS_TYPE_PTR pnFsType
);
```

File

PrlApiDisp.h

Parameters

hFsInfo

A handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206) identifying the remote file system entry.

pnFsType

[out] A pointer to a variable that receives the file entry type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsInfo_GetParentEntry Function

Obtains a handle of type PHT_REMOTE_FILESYSTEM_ENTRY (p. 202) to the parent file system entry.

Syntax

```
PRL_RESULT PrlFsInfo_GetParentEntry(
    PRL_HANDLE hFsInfo,
    PRL_HANDLE_PTR phParentEntry
);
```

File

PrlApiDisp.h

Parameters

hFsInfo

A handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206) identifying the remote file system entry.

phParentEntry

[out] A pointer to a variable that receives the handle. The handle type is PHT_REMOTE_FILESYSTEM_ENTRY (p. 202).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlFsInfo_GetType Function

Determines the basic type of the specified file system entry.

Syntax

```
PRL_RESULT PrlFsInfo_GetType(
    PRL_HANDLE hFsInfo,
    PRL_FILE_SYSTEM_TYPE_PTR pnFsType
);
```

File

PrlApiDisp.h

Parameters

hFsInfo

A handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206) identifying the remote file system entry.

pnFsType

[out] A pointer to a variable that receives the file entry type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_RESULT

The PHT_RESULT handle is a container that contains results of an asynchronous operation. The handle is obtained from the PHT_JOB (p. 138) object returned by the asynchronous function.

Description

To obtain a handle, use the PrlJob_GetResult (p. 142) function call after the job completes. Once you have the handle, use functions from this group to extract the results of the corresponding asynchronous operation. The PHT_RESULT object can contain a string, an array of strings, a handle, or an array of handles.

If you are expecting a single result, then, depending on the result type (a string or a handle), use the PrlResult_GetParamAsString (p. 213) or the PrlResult_GetParam (p. 210) function respectively.

If the function returns multiple results, first, use the PrlResult_GetParamsCount (p. 216) function to get the number of the results. Then use the PrlResult_GetParamByIndex (p. 213) or the PrlResult_GetParamByIndexAsString (p. 215) function to get individual results. The index of the array begins at 0.

For more information, please also see the **Virtuozzo Virtualization SDK Programmer's Guide**.

Functions

	Name	Description
	PrlResult_GetParam (p. 210)	Obtains a handle to an object containing the results of the corresponding asynchronous operation.
	PrlResult_GetParamAsString (p. 213)	Obtains a string result from the result object.
	PrlResult_GetParamByIndex (p. 213)	Obtains a handle from the result object identified by the index.
	PrlResult_GetParamByIndexAsString (p. 215)	Obtains a string result from the result object identified by the index.
	PrlResult_GetParamsCount (p. 216)	Determines the number of items in the specified result object.

PrlResult_GetParam Function

Obtains a handle to an object containing the results of the corresponding asynchronous operation.

Syntax

```
PRL_RESULT PrlResult_GetParam(
    PRL_HANDLE hResult,
    PRL_HANDLE_PTR pHandle
);
```

File

PrlApiCore.h

Parameters

hResult

A handle of type PHT_RESULT (p. 210) identifying the result object.

pHandle

[out] A pointer to a variable that receives the handle. The type of handle returned depends on the corresponding operation. The description of each asynchronous function in this documentation provides information about the type of handle that the function returns.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - no result or the result is not a single handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when you know that the corresponding function always produces a single object as a result. This function is the equivalent of the PrlResult_GetParamByIndex (p. 213)(hResult, 0, &Handle) call (i.e. zero as the index value).

Example

The following example demonstrates using asynchronous function PrlSrv_GetSrvConfig (p. 257) synchronously, and the steps involved in obtaining a handle to an object of type PHT_SERVER_CONFIG (p. 314) (which is returned by PrlSrv_GetSrvConfig (p. 257)):

```
PRL_HANDLE hJob = PrlSrv_GetSrvConfig(hServer);

// Step 1 - emulate a synchronous call using PrlJob_Wait.

PRL_RESULT ret = PrlJob_Wait(hJob, 10000);
if (PRL_FAILED(ret))
{
    printf("PrlJob_Wait for job PrlSrv_GetSrvConfig returned error: %s\n",
        PRL_RESULT_TO_STRING(ret));

    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Step 2 - determine if PrlSrv_GetSrvConfig succeeded or failed
// using PrlJob_GetRetCode.

PrlJob_GetRetCode(hJob, &nJobResult);
if (PRL_FAILED(nJobResult))
{
    printf("PrlSrv_GetSrvConfig returned error: %s\n",
        PRL_RESULT_TO_STRING(ret));

    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Step 3 - Obtain a handle to the result object (a handle
// of type PHT_RESULT) using PrlJob_GetResult.

PRL_HANDLE hResult;
ret = PrlJob_GetResult(hJob, &hResult);
PrlHandle_Free(hJob); // hJob no longer needed, free it.
```

```
if (PRL_FAILED(ret))
{
    printf("PrlJob_GetResult for job PrlSrv_GetSrvConfig returned error: %s\n",
        PRL_RESULT_TO_STRING(ret));
    PrlHandle_Free(hResult);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Step 4 - Get a handle to the final result (in this case,
// a handle of type PHT_SERVER_CONFIG) using PrlResult_GetParam.
// For PHT_RESULT handles that contain more than one result, first use
// PrlResult_GetParamsCount, then use PrlResult_GetParamByIndex.

PRL_HANDLE hServerConfig;
ret = PrlResult_GetParam(hResult, &hServerConfig);
PrlHandle_Free(hResult); // hResult no longer needed, free it.

if (PRL_FAILED(ret))
{
    printf("PrlResult_GetParam for job PrlSrv_GetSrvConfig returned error: %s\n",
        PRL_RESULT_TO_STRING(ret));

    PrlHandle_Free(hServerConfig);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();

    return -1;
}

// At this point, a handle of type PHT_SERVER_CONFIG is available, and
// functions that operate on PHT_SERVER_CONFIG handles can be used on
// the handle.

PRL_UINT32 nCpuCount = 0;
PrlSrvCfg_GetCpuCount(hServerConfig, &nCpuCount);
printf("Number of CPUs: %d\n", nCpuCount);

PRL_CPU_MODE CpuMode;
PrlSrvCfg_GetCpuMode(hServerConfig, &CpuMode);
printf("CPU Mode: %d bit\n", CpuMode == PCM_CPU_MODE_32 ? 32 : 64);

PRL_CHAR szCpuModel[1024];
PRL_UINT32 nCpuModelSize = sizeof(szCpuModel);
memset(szCpuModel, 0, nCpuModelSize);
PrlSrvCfg_GetCpuModel(hServerConfig, szCpuModel, &nCpuModelSize);
printf("CPU Model: %s\n", szCpuModel);

PRL_UINT32 nCpuSpeed = 0;
PrlSrvCfg_GetCpuSpeed(hServerConfig, &nCpuSpeed);
printf("CPU Speed: %.3f MHz\n", nCpuSpeed / 1000.0);

PRL_CHAR szHostOsString[1024];
PRL_UINT32 nHostOsStringSize = sizeof(szHostOsString);
PrlSrvCfg_GetHostOsStrPresentation(hServerConfig, szHostOsString, &nHostOsStringSize);
printf("Host OS: %s\n", szHostOsString);

PrlHandle_Free(hServerConfig);
```

PrlResult_GetParamAsString Function

Obtains a string result from the result object.

Syntax

```
PRL_RESULT PrlResult_GetParamAsString(
    PRL_HANDLE hResult,
    PRL_STR sParamBuf,
    PRL_UINT32_PTR pnParamBufSize
);
```

File

PrlApiCore.h

Parameters

hResult

A handle of type PHT_RESULT (p. 210) identifying the result object.

sParamBuf

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnParamBufSize

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when you know that the corresponding function always produces a single string as a result. This function is the equivalent of the PrlResult_GetParamByIndexAsString (p. 215)(hResult, 0, &myString) call (i.e. zero as the index value).

PrlResult_GetParamByIndex Function

Obtains a handle from the result object identified by the index.

Syntax

```
PRL_RESULT PrlResult_GetParamByIndex(
    PRL_HANDLE hResult,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phParam
);
```

File

PrlApiCore.h

Parameters

nIndex

An index identifying the handle in the list.

phParam

[out] A pointer to a variable that receives the handle. The type of handle returned depends on the corresponding operation. The description of each asynchronous function in this documentation provides information about the type of handle that the function returns.

handle

A handle of type PHT_RESULT (p. 210) identifying the result object.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when the result contains (or may contain) multiple objects. To obtain multiple handles from the result object, first, determine the number of handles contained in the object using the PrlResult_GetParamsCount (p. 216) function. Then, iterate through the list and use the iteration number as the value of the nIndex parameter.

Example

The following example demonstrates using asynchronous function PrlSrv_GetVmList (p. 264) (getting the list of virtual machines), and the steps involved in obtaining handles to objects of type PHT_VIRTUAL_MACHINE (p. 442) returned by PrlSrv_GetVmList (p. 264) (each object contains information about an individual virtual machine):

```
PRL_RESULT DisplayVmList(const PRL_HANDLE &hServer)
{
    PRL_HANDLE hResult;
    PRL_RESULT nJobResult;

    // Get a list of virtual machines.
    PRL_HANDLE hJob = PrlSrv_GetVmList(hServer);

    // Wait for the job to complete.
    PRL_RESULT ret = PrlJob_Wait(hJob, 10000);
    if (PRL_FAILED(ret))
    {
        fprintf(stderr, "PrlJob_Wait for PrlSrv_GetVmList returned with error: %s\n",
            PRL_RESULT_TO_STRING(ret));
        PrlHandle_Free(hJob);
        return ret;
    }

    // Check the results of PrlSrv_GetVmList.
    ret = PrlJob_GetRetCode(hJob, &nJobResult);
    if (PRL_FAILED(nJobResult))
    {
        fprintf(stderr, "PrlSrv_GetVmList returned with error: %s\n",
            PRL_RESULT_TO_STRING(ret));
        PrlHandle_Free(hJob);
        return ret;
    }

    // Get the results of PrlSrv_GetVmList.
    ret = PrlJob_GetResult(hJob, &hResult);
    if (PRL_FAILED(ret))
    {
```

```
            fprintf(stderr, "PrlJob_GetResult returned with error: %s\n",
                PRL_RESULT_TO_STRING(ret));
            PrlHandle_Free(hResult);
            PrlHandle_Free(hJob);
            return ret;
        }

        // Handle to result object is available,
        // job handle is not longer needed, so free it.
        PrlHandle_Free(hJob);

        // Iteratre through the result list.
        PRL_UINT32 nParamsCount = 0;
        ret = PrlResult_GetParamsCount(hResult, &nParamsCount);

        for (PRL_UINT32 i = 0; i < nParamsCount; ++i)
        {
            PRL_HANDLE hVm;

            // Get a handle to result at index i.
            PrlResult_GetParamByIndex(hResult, i, &hVm);

            // Get the name of the virtual machine for result i.
            char szVmNameReturned[1024];
            PRL_UINT32 nBufSize = sizeof(szVmNameReturned);
            ret = PrlVm_GetName(hVm, szVmNameReturned, &nBufSize);

            if (PRL_FAILED(ret))
            {
                printf("PrlVm_GetName returned with error (%s)\n",
                    PRL_RESULT_TO_STRING(ret));
            }
            else
            {
                printf("Virtual machine '%s' is available.\n\n", szVmNameReturned);
            }

            // Free the virtual machine handle.
            PrlHandle_Free(hVm);
        }

        return PRL_ERR_SUCCESS;
    }
```

PrlResult_GetParamByIndexAsString Function

Obtains a string result from the result object identified by the index.

Syntax

```
PRL_RESULT PrlResult_GetParamByIndexAsString(
    PRL_HANDLE hResult,
    PRL_UINT32 nIndex,
    PRL_STR sParamBuf,
    PRL_UINT32_PTR pnParamBufSize
);
```

File

PrlApiCore.h

Parameters

hResult

A handle of type PHT_RESULT (p. 210) identifying the result object.

nIndex

An index identifying the string item in the list.

sParamBuf

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnParamBufSize

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when the result contains (or may contain) multiple string items. To obtain multiple items from the result object, first, determine the number of items contained in the object using the PrlResult_GetParamsCount (p. 216) function. Then, iterate through the list and use the iteration number as the value of the nIndex parameter.

PrlResult_GetParamsCount Function

Determines the number of items in the specified result object.

Syntax

```
PRL_RESULT PrlResult_GetParamsCount(
    PRL_HANDLE hResult,
    PRL_UINT32_PTR pCount
);
```

File

PrlApiCore.h

Parameters

hResult

A handle of type PHT_RESULT (p. 210) identifying the result object.

pCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_RUNNING_TASK

The PHT_RUNNING_TASK handle is used to recover from the lost Virtuozzo Service connection. It allows to attach to an existing task that was started in the previous session and is still running inside the Virtuozzo Service. The handle is obtained using the PrlLoginResponse_GetRunningTaskByIndex (p. 151) function. Only a limited number of tasks is supported by this functionality. They are defined in the PRL_RUNNING_TASK_TYPE_ENUM (p. 1090) enumeration. For more information, please also see the PrlSrv_AttachToLostTask (p. 224) function.

Functions

	Name	Description
	PrlRunningTask_GetTaskParametersAsString (p. 217)	Returns task parameters as a string.
	PrlRunningTask_GetTaskType (p. 218)	Determines the task type.
	PrlRunningTask_GetTaskUuid (p. 218)	Returns the task UUID (universally unique ID).

PrlRunningTask_GetTaskParametersAsString Function

Returns task parameters as a string.

Syntax

```
PRL_RESULT PrlRunningTask_GetTaskParametersAsString(
    PRL_HANDLE hRunningTask,
    PRL_STR sParamsStr,
    PRL_UINT32_PTR pnsParamsStrBufLength
);
```

File

PrlApiDisp.h

Parameters

hRunningTask

A handle of type PHT_RUNNING_TASK (p. 217) identifying the task.

sParamsStr

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnsParamsStrBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Value returns in UTF-8 encoding. Null char ('0') adding to the end of string.

PrlRunningTask_GetTaskType Function

Determines the task type.

Syntax

```
PRL_RESULT PrlRunningTask_GetTaskType(
    PRL_HANDLE hRunningTask,
    PRL_RUNNING_TASK_TYPE_ENUM_PTR pnType
);
```

File

PrlApiDisp.h

Parameters

hRunningTask

A handle of type PHT_RUNNING_TASK (p. 217) identifying the task.

pnType

[out] A pointer to a variable that receives the task type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlRunningTask_GetTaskUuid Function

Returns the task UUID (universally unique ID).

Syntax

```
PRL_RESULT PrlRunningTask_GetTaskUuid(
    PRL_HANDLE hRunningTask,
    PRL_STR sTaskUuid,
    PRL_UINT32_PTR pnTaskUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hRunningTask

A handle of type PHT_RUNNING_TASK (p. 217) identifying the task.

sTaskUuid

[out] A pointer to a buffer that receives the task UUID (a UTF-8 encoded, null-terminated string).

pnTaskUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_SCREEN_RESOLUTION

The PHT_SCREEN_RESOLUTION handle is used to modify the list of screen resolutions available in a virtual machine. By default, only the most common resolutions are supported in a virtual machine. Using this handle, you can add additional resolutions and modify the existing ones if needed. The handle is obtained using the PrlVm_CreateScrRes or the PrlVm_GetScrRes function. The additional screen resolution support can be enabled or disabled in a virtual machine. To determine the status of the feature, use the PrlVmCfg_IsScrResEnabled (p. 636) function.

PHT_SERVER

At the root of the Virtuozzo API class hierarchy is a handle of type PHT_SERVER. The handle can be obtained using the PrlSrv_Create (p. 230) function. The PHT_SERVER handle is required to use most of the functionality within the Virtuozzo API.

Functions

	Name	Description
	PrlSrv_AddIPPrivateNetwork (p. 223)	Adds a new private network to the Virtuozzo Service configuration.
	PrlSrv_AddVirtualNetwork (p. 224)	Adds a new virtual network to the Virtuozzo Service configuration.
	PrlSrv_AttachToLostTask (p. 224)	The PrlSrv_AttachToLostTask function allows to obtain a handle to a running task after the connection to the Virtuozzo Service was lost.
	PrlSrv_CancelInstallAppliance (p. 225)	Cancels the appliance installation previously started with the PrlSrv_InstallAppliance (p. 267) function call.
	PrlSrv_CommonPrefsBeginEdit (p. 226)	The PrlSrv_CommonPrefsBeginEdit function is used in modifying Virtuozzo Service preferences (see PHT_DISP_CONFIG (p. 32)).
	PrlSrv_CommonPrefsCommit (p. 227)	The PrlSrv_CommonPrefsCommit function is used in modifying Virtuozzo Service preferences (see PHT_DISP_CONFIG (p. 32)).
	PrlSrv_CommonPrefsCommitEx (p. 228)	
	PrlSrv_ConfigureGenericPci (p. 228)	Configures the PCI device assignment.
	PrlSrv_CopyCtTemplate (p. 229)	Copy the specified Containers template to the specified server.
	PrlSrv_Create (p. 230)	Creates a new handle of type PHT_SERVER.
	PrlSrv_CreateUnattendedCd (p. 231)	Creates a bootable ISO-image for unattended Linux installation.

	PrlSrv_CreateVm (p. 232)	Obtains a new handle of type PHT_VIRTUAL_MACHINE (p. 442).
	PrlSrv_CreateVmBackup (p. 233)	Backup an existing virtual machine to backup server.
	PrlSrv_DeleteOfflineService (p. 234)	Removes an existing offline service configuration.
	PrlSrv_DeleteVirtualNetwork (p. 235)	Removes an existing virtual network from the Virtuozzo Service configuration.
	PrlSrv_DisableConfirmationMode (p. 236)	Disable administrator confirmation mode for the session.
	PrlSrv_EnableConfirmationMode (p. 236)	Enable administrator confirmation mode for the session.
	PrlSrv_FsCanCreateFile (p. 237)	Determines if the current user has sufficient rights to create a file or a directory in the specified location on the host.
	PrlSrv_FsCreateDir (p. 238)	Creates a directory in the specified location on the host.
	PrlSrv_FsGenerateEntryName (p. 238)	Automatically generates a unique name for a new directory.
	PrlSrv_FsGetDirEntries (p. 239)	Retrieves information about the specified file system entry residing on the specified host computer.
	PrlSrv_FsGetDiskList (p. 240)	Returns a list of root directories on the host computer.
	PrlSrv_FsRemoveEntry (p. 241)	Removes a file system entry from the host computer.
	PrlSrv_FsRenameEntry (p. 242)	Renames a file system entry on the host.
	PrlSrv_GetBackupTree (p. 242)	Get backups tree from backup server.
	PrlSrv_GetCommonPrefs (p. 244)	Returns a handle of type PHT_DISP_CONFIG (p. 32) containing the specified Virtuozzo Service preferences information.
	PrlSrv_GetCPUPoolsList (p. 245)	Obtains a list of handles of type PHT_CPU_POOL (p. 24) containing information about all existing cpu pools.
	PrlSrv_GetCtTemplateList (p. 246)	Retrieves a list of Containers templates registered with the specified Virtuozzo Service.
	PrlSrv_GetDefaultVmConfig (p. 246)	Obtains a new handle of type PHT_VIRTUAL_MACHINE (p. 442).
	PrlSrv_GetDiskFreeSpace (p. 247)	Get available disk space for path.
	PrlSrv_GetIPPrivateNetworksList (p. 248)	Obtains a list of handles of type PHT_IPPRIV_NET (p. 133) containing information about all existing private networks.
	PrlSrv_GetLicenseInfo (p. 249)	Returns an object of type PHT_LICENSE (p. 146) containing the Virtuozzo license information.
	PrlSrv_GetNetServiceStatus (p. 249)	Returns an object of type PHT_NET_SERVICE_STATUS (p. 153) containing the Virtuozzo network service status information.
	PrlSrv_GetNetworkClassesList (p. 250)	Obtains a list of handles of type PHT_HANDLES_LIST (p. 108) containing the list of handles of type PHT_NETWORK_CLASS (p. 154).
	PrlSrv_GetNetworkShapingConfig (p. 251)	Obtains the network shaping configuration.
	PrlSrv_GetOfflineServicesList (p. 252)	Obtains a list of handles of type PHT_OFFLINE_SERVICE (p. 173) containing information about all existing services.
	PrlSrv_GetPackedProblemReport (p. 252)	Allows to receive a packed problem report from the Virtuozzo Service in the event of a virtual machine operation failure.
	PrlSrv_GetPerfStats (p. 253)	Obtains a handle of type PHT_EVENT (p. 72) containing the perfomance statistics.
	PrlSrv_GetPluginsList (p. 254)	Gets plugin list with plugin info entries.
	PrlSrv_GetProblemReport (p. 254)	Allows to receive a problem report from the Virtuozzo Service in the event of a virtual machine operation failure.
	PrlSrv_GetQuestions (p. 255)	The PrlSrv_GetQuestions function allows to synchronously receive questions from a Virtuozzo Service.
	PrlSrv_GetRestrictionInfo (p. 256)	Gets restriction info.

	PrlSrv_GetServerInfo (p. 256)	Obtains a handle of type PHT_SERVER_INFO (p. 344) containing the host computer information.
	PrlSrv_GetSrvConfig (p. 257)	Obtains a handle of type PHT_SERVER_CONFIG (p. 314) containing the host configuration information.
	PrlSrv_GetStatistics (p. 258)	Obtains a handle of type PHT_SYSTEM_STATISTICS (p. 360) containing the host resource usage statistics.
	PrlSrv_GetSupportedOses (p. 259)	Extracts supported guest OSes matrix from remote server (it's more proper way instead of using local synch API in case when client and server interoperate remotely).
	PrlSrv_GetUserInfo (p. 260)	Obtains a handle of type PHT_USER_INFO (p. 371) containing information about the specified user.
	PrlSrv_GetUserInfoList (p. 261)	Obtains a list of handles of type PHT_USER_INFO (p. 371) containing information about all known users.
	PrlSrv_GetUserProfile (p. 262)	Obtains a handle of type PHT_USER_PROFILE (p. 374) containing the profile data of the currently logged in user.
	PrlSrv_GetVcmmdConfig (p. 262)	Obtain a handle to vcmmd config.
	PrlSrv_GetVirtualNetworkList (p. 263)	Obtains a list of handles of type PHT_VIRTUAL_NETWORK (p. 507) containing information about all existing virtual networks.
	PrlSrv_GetVmConfig (p. 264)	Get VM config by given UUID or name.
	PrlSrv_GetVmList (p. 264)	Retrieves a list of virtual machines registered with the specified Virtuozzo Service.
	PrlSrv_GetVmListEx (p. 266)	Retrieves a list of virtual machines registered with the specified Virtuozzo Service.
	PrlSrv_HasRestriction (p. 266)	Determines having restriction.
	PrlSrv_InstallAppliance (p. 267)	Sends an appliance configuration to the Virtuozzo Service in order to install the corresponding appliance from Virtuozzo download service.
	PrlSrv_IsConfirmationModeEnabled (p. 268)	Determines confirmation mode for the session
	PrlSrv_IsConnected (p. 268)	Determines if the connection to the specified Virtuozzo Service is active.
	PrlSrv_IsFeatureSupported (p. 269)	Checks whether specified feature supported by remote side.
	PrlSrv_IsNonInteractiveSession (p. 270)	Determines if the session is noninteractive or interactive.
	PrlSrv_IsProxyConnection (p. 270)	Determines if the connection to the specified Virtuozzo Service is a proxy connection.
	PrlSrv_JoinCPUPool (p. 271)	Join server to the default cpu pool To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.
	PrlSrv_LeaveCPUPool (p. 271)	Leaves server from any cpu pool To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.
	PrlSrv_Login (p. 272)	Performs a remote login operation using the specified parameters.
	PrlSrv_LoginEx (p. 276)	
	PrlSrv_LoginLocal (p. 276)	Allows to log in to the local Virtuozzo Service (the Virtuozzo Service running on this host) using the current user credentials.
	PrlSrv_LoginLocalEx (p. 278)	
	PrlSrv_Logoff (p. 278)	Logs off the Virtuozzo Service.
	PrlSrv_MoveToCPUPool (p. 280)	Moves server to a different cpu pool To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.
	PrlSrv_NetServiceRestart (p. 281)	Restarts the Virtuozzo network service.

	PrlSrv_NetServiceRestoreDefaults (p. 281)	Restores the default settings of the Virtuozzo network service.
	PrlSrv_NetServiceStart (p. 282)	Starts the Virtuozzo network service.
	PrlSrv_NetServiceStop (p. 282)	Stops the Virtuozzo network service.
	PrlSrv_RecalculateCPUPool (p. 283)	Recalculates features in pool To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.
	PrlSrv_RefreshPlugins (p. 283)	Refreshes states of installed Virtuozzo plugins.
	PrlSrv_RegEventHandler (p. 284)	Registers an event handler.
	PrlSrv_Register3rdPartyVm (p. 285)	Registers an existing 3rd party virtual machine with Virtuozzo Service.
	PrlSrv_RegisterVm (p. 286)	Registers an existing virtual machine with Virtuozzo Service.
	PrlSrv_RegisterVmEx (p. 288)	Registers an existing virtual machine with Virtuozzo Service.
	PrlSrv_RegisterVmWithUuid (p. 289)	Registers an existing virtual machine with custom Vm UUID.
	PrlSrv_RemoveCtTemplate (p. 290)	Removes the specified Containers template.
	PrlSrv_RemoveIPPrivateNetwork (p. 291)	Removes an existing private network from the Virtuozzo Service configuration.
	PrlSrv_RemoveVmBackup (p. 292)	Remove backup of the virtual machine from backup server.
	PrlSrv_RestoreVmBackup (p. 293)	Restore backup of the virtual machine from backup server.
	PrlSrv_SendAnswer (p. 294)	Sends an answer to the Virtuozzo Service in response to a question.
	PrlSrv_SendProblemReport (p. 295)	Requests a server to send a problem report with possibility to append also the problem report assemblied at the client's side.
	PrlSrv_SetNonInteractiveSession (p. 296)	Sets the session in noninteractive or interactive mode.
	PrlSrv_SetVcmmdConfig (p. 296)	Set handle of vcmmd config.
	PrlSrv_SetVNCEncryption (p. 297)	Set VNC encryption.
	PrlSrv_Shutdown (p. 297)	Shuts down the Virtuozzo Service.
	PrlSrv_ShutdownEx (p. 298)	
	PrlSrv_StartSearchVms (p. 299)	Searches for unregistered virtual machines at the specified location(s).
	PrlSrv_StopInstallAppliance (p. 300)	Stops the appliance installation previously started with the PrlSrv_InstallAppliance (p. 267) function call.
	PrlSrv_SubscribeToHostStatistics (p. 301)	Allows to subscribe to receive host OS statistics on a periodic basis.
	PrlSrv_SubscribeToPerfStats (p. 305)	Allows to subscribe to periodically receive perfomance statistics.
	PrlSrv_UnregEventHandler (p. 306)	Unregisters the specified event handler (callback function) that was previously registered with the PrlSrv_RegEventHandler (p. 284) function.
	PrlSrv_UnsubscribeFromHostStatistics (p. 307)	Cancels the Virtuozzo Service statistics subscription that was started using the PrlSrv_SubscribeToHostStatistics (p. 301) function.
	PrlSrv_UnsubscribeFromPerfStats (p. 307)	Cancels the Virtuozzo Service perfomace statistics subscription that was started using the PrlSrv_SubscribeToPerfStats (p. 305) function.
	PrlSrv_UpdateIPPrivateNetwork (p. 308)	Updates parameters of an existing private network.
	PrlSrv_UpdateLicense (p. 309)	Installs Virtuozzo license on the specified Virtuozzo Service.
	PrlSrv_UpdateLicenseEx (p. 310)	
	PrlSrv_UpdateNetworkClassesList (p. 310)	Update network class classes configuration.
	PrlSrv_UpdateNetworkShapingConfig (p. 311)	Update the network shaping configuration.
	PrlSrv_UpdateOfflineService (p. 311)	Add/Updates parameters of an offline management service.

	PrlSrv_UpdateVirtualNetwork (p. 312)	Updates parameters of an existing virtual network.
	PrlSrv_UserProfileBeginEdit (p. 313)	Timestamps the beginning of the user profile editing operation.
	PrlSrv_UserProfileCommit (p. 314)	Saves (commits) user profile changes to the Virtuozzo Service.

Topics

Name	Description
PrlSrv_CheckParallelsServerAlive@PRL_UINT32@PRL_CONST_STR@PRL_BOOL_PTR@PRL_HANDLE_PTR (p. 226)	
PrlSrv_LookupParallelsServers@PRL_UINT32@PRL_EVENT_HANDLER_PTR@PRL_VOID_PTR (p. 279)	

PrlSrv_AddIPPrivateNetwork Function

Adds a new private network to the Virtuozzo Service configuration.

Syntax

```
PRL_HANDLE PrlSrv_AddIPPrivateNetwork(
    PRL_HANDLE hServer,
    PRL_HANDLE hPrivNet,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying the private network to add to the Virtuozzo Service configuration.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To create a new private network, first create a handle of type PHT_IPPRIV_NET (p. 133), configure the network using the handle's functions, and then call this function to add the network to the Virtuozzo Service configuration.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_GetIPPrivateNetworkList

PrlSrv_DeleteIPPrivateNetwork

PrlSrv_UpdateIPPrivateNetwork (p. 308)

PrlSrv_AddVirtualNetwork Function

Adds a new virtual network to the Virtuozzo Service configuration.

Syntax

```
PRL_HANDLE PrlSrv_AddVirtualNetwork(
    PRL_HANDLE hServer,
    PRL_HANDLE hVirtNet,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network to add to the Virtuozzo Service configuration.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To create a new virtual network, first create a handle of type PHT_VIRTUAL_NETWORK (p. 507), configure the network using the handle's functions, and then call this function to add the network to the Virtuozzo Service configuration.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_GetVirtualNetworkList (p. 263)

PrlSrv_DeleteVirtualNetwork (p. 235)

PrlSrv_UpdateVirtualNetwork (p. 312)

PrlSrv_AttachToLostTask Function

The PrlSrv_AttachToLostTask function allows to obtain a handle to a running task after the connection to the Virtuozzo Service was lost.

Syntax

```
PRL_HANDLE PrlSrv_AttachToLostTask(
    PRL_HANDLE hServer,
    PRL_CONST_STR sTaskId
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sTaskId

The ID of the task to get a job handle for.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object. The returned PHT_JOB (p. 138) object will be attached to the specified task.

Remarks

If you've started a long asynchronous operation and then lost a connection with the Virtuozzo Service, you may use this function to get a handle to the task after reconnecting to the Virtuozzo Service. In order to do that, log in to the Virtuozzo Service using the PrlSrv_Login (p. 272) or the PrlSrv_LoginLocal (p. 276) function passing the UUID of the session that you've lost. The returned PHT_LOGIN_RESPONSE (p. 149) object will contain the list of the tasks that belong to the lost session and are still running. Select a task that you want to get a handle to and pass its ID to the PrlSrv_AttachToLostTask function. The function will return a handle of type PHT_JOB (p. 138) which you may use to continue controlling and monitoring the task as usual.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_CancelInstallAppliance Function

Cancels the appliance installation previously started with the PrlSrv_InstallAppliance (p. 267) function call.

Syntax

```
PRL_HANDLE PrlSrv_CancelInstallAppliance(
    PRL_HANDLE hServer,
    PRL_HANDLE hAppCfg,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hAppCfg

A handle of type PHT_APPLIANCE_CONFIG identifying the appliance. This must be the same handle instance that was passed to the PrlSrv_InstallAppliance (p. 267) function.

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_CheckParallelsServerAlive@PRL_UINT32@PRL_CONST_STR@PRL_BOOL_PTR@PRL_HANDLE_PTR

PrlSrv_CommonPrefsBeginEdit Function

The PrlSrv_CommonPrefsBeginEdit function is used in modifying Virtuozzo Service preferences (see PHT_DISP_CONFIG (p. 32)).

Syntax

```
PRL_HANDLE PrlSrv_CommonPrefsBeginEdit(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The PrlSrv_CommonPrefsBeginEdit and the PrlSrv_CommonPrefsCommit (p. 227) functions are used to detect collisions with other clients trying to modify preferences of the same Virtuozzo Service. The PrlSrv_CommonPrefsBeginEdit call timestamps the beginning of the operation. It does not lock the Virtuozzo Service, so other clients can modify the same Virtuozzo Service settings at the same time. When you are done making the changes, you must call the PrlSrv_CommonPrefsCommit (p. 227) function to apply them to the Virtuozzo Service. The function will verify that the Virtuozzo Service preferences have not been

modified by another client. If a collision is detected, your changes will be rejected.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_ACCESS_DENIED (p. 1175) - current user does not have sufficient rights to perform this operation.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_CommonPrefsCommit Function

The PrlSrv_CommonPrefsCommit function is used in modifying Virtuozzo Service preferences (see PHT_DISP_CONFIG (p. 32)).

Syntax

```
PRL_HANDLE PrlSrv_CommonPrefsCommit(
    PRL_HANDLE hServer,
    PRL_HANDLE hDispCfg
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hDispCfg

A handle of type PHT_DISP_CONFIG (p. 32) containing the Virtuozzo Service preferences.

nFlags

Reserved flags

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The PrlSrv_CommonPrefsBeginEdit (p. 226) and the PrlSrv_CommonPrefsCommit functions are used to detect collisions with other clients trying to modify preferences of the same Virtuozzo Service. The PrlSrv_CommonPrefsBeginEdit (p. 226) call timestamps the beginning of the operation. It does not lock the Virtuozzo Service, so other clients can modify the same Virtuozzo Service at the same time. When you are done making the changes, you must call the PrlSrv_CommonPrefsCommit function to apply them to the Virtuozzo Service. The function will verify that the Virtuozzo Service preferences have not been modified by another client. If a collision is detected, your changes will be rejected.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_CommonPrefsCommitEx Function

Syntax

```
PRL_HANDLE PrlSrv_CommonPrefsCommitEx(
    PRL_HANDLE hServer,
    PRL_HANDLE hDispCfg,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

PrlSrv_ConfigureGenericPci Function

Configures the PCI device assignment.

Syntax

```
PRL_HANDLE PrlSrv_ConfigureGenericPci(
    PRL_HANDLE hServer,
    PRL_HANDLE hDevList,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hDevList

A handle of type PHT_HANDLES_LIST (p. 108) containing a list of handles of type PHT_HW_GENERIC_PCI_DEVICE (p. 115). Each handle in the list identifies an individual PCI device installed on the host computer.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

For more information about generic PCI devices and their usage, see the PHT_HW_GENERIC_PCI_DEVICE (p. 115) handle description.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PHT_HW_GENERIC_PCI_DEVICE (p. 115)

PrlSrv_CopyCtTemplate Function

Copy the specified Containers template to the specified server.

Syntax

```
PRL_HANDLE PrlSrv_CopyCtTemplate(
    PRL_HANDLE hSrv,
    PRL_CONST_STR sName,
    PRL_CONST_STR sOsTmplName,
    PRL_CONST_STR sTargetServerHostname,
    PRL_UINT32 nTargetServerPort,
    PRL_CONST_STR sTargetServerSessionUuid,
    PRL_UINT32 nFlags,
    PRL_UINT32 nReservedFlags
);
```

File

PrlApiVm.h

Parameters

hSrv

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sName

Name of template to be removed.

sOsTmplName

Name of OS template for which this application template is. Should be filled for application templates.

sTargetServerHostname

target server hostname

nTargetServerPort

target server port

sTargetServerSessionUuid

target server session uuid

nFlags

Command flags.

nReservedFlags

Reserved command flags.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_Create Function

Creates a new handle of type PHT_SERVER (p. 219).

Syntax

```
PRL_RESULT PrlSrv_Create(
    PRL_HANDLE_PTR handle
);
```

File

PrlApiDisp.h

Parameters

handle

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

In order to establish a connection with a Virtuozzo Service, a handle of type PHT_SERVER (p. 219) must be created and the object that the handle references must be populated with the Virtuozzo Service connection information. Once a connection is established, the handle can be used to access the Virtuozzo Service. It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

See Also

PrlSrv_LoginLocal (p. 276)

PrlSrv_Login (p. 272)

Example

The following example illustrates how to create a handle of type PHT_SERVER (p. 219).

```
PRL_HANDLE hServer;

PRL_RESULT ret = PrlSrv_Create( &hServer );
if ( PRL_FAILED(ret) )
{
    printf( "PrlSrv_Create returned with error: %s\n",
    PRL_RESULT_TO_STRING(ret) );
}

// Free hServer handle when it's no longer needed.
PrlHandle_Free( hServer );
```

PrlSrv_CreateUnattendedCd Function

Creates a bootable ISO-image for unattended Linux installation.

Syntax

```
PRL_HANDLE PrlSrv_CreateUnattendedCd(
    PRL_HANDLE hServer,
    PRL_UINT32 nGuestType,
    PRL_CONST_STR sUserName,
    PRL_CONST_STR sPasswd,
    PRL_CONST_STR sFullUserName,
    PRL_CONST_STR sOsDistroPath,
    PRL_CONST_STR sOutImagePath
);
```

File

PrlApiVm.h

Parameters

hServer
A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sUserName
User name.

sPasswd
Password.

sFullUserName
Full user name.

sOsDistroPath
A full path to the OS distribution image or folder.

sOutImagePath
A full path to output ISO-image file.

nGuestDistroType
Guest OS version from the (PVS_GUEST_VER_...).

sPath
Full path to the OS distribution image or folder.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The following steps describe how to prepare a virtual machine for unattended Linux installation:

1. Create a virtual machine. The machine should have all the necessary devices, like hard drive and CD drive.
2. Insert Linux installation CD-ROM into the drive (or mount a disk image).
3. Execute the PrlVm_CreateUnattendedCd function to create a bootable ISO-image.

4. Set created ISO-image as boot device.
5. Start the virtual machine. The machine will boot from the created ISO-image.
6. Perform unattended Windows installation as usual.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_CreateVm Function

Obtains a new handle of type PHT_VIRTUAL_MACHINE (p. 442).

Syntax

```
PRL_RESULT PrlSrv_CreateVm(
    PRL_HANDLE hServer,
    PRL_HANDLE_PTR phVm
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

phVm

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Obtaining a new PHT_VIRTUAL_MACHINE (p. 442) handle is the first step when creating a virtual machine. The second step is setting the virtual machine configuration parameters. The final step is calling the PrlVm_Reg (p. 483) function, which will create a new virtual machine.

Example

```
// Create a new virtual machine handle
// (assumes hServer is a valid handle obtained using PrlSrv_Create,
// and a successful Virtuozzo Service login was made using PrlSrv_Login).
PRL_HANDLE hNewVm;
ret = PrlSrv_CreateVm(hServer, &hNewVm);
if (PRL_FAILED(ret))
{
    printf("PrlSvr_CreateVm call failed with: %s.\n", PRL_RESULT_TO_STRING(ret));
    return ret;
}
printf("PrlSvr_CreateVm completed successfully.\n");
```

PrlSrv_CreateVmBackup Function

Backup an existing virtual machine to backup server.

Syntax

```
PRL_HANDLE PrlSrv_CreateVmBackup(
    PRL_HANDLE hSourceServer,
    PRL_CONST_STR sVmUuid,
    PRL_CONST_STR sTargetHost,
    PRL_UINT32 nTargetPort,
    PRL_CONST_STR sTargetSessionId,
    PRL_CONST_STR strDescription,
    PRL_UINT32 backup_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hSourceServer

A handle of type PHT_SERVER (p. 219) identifying the source Virtuozzo Service.

sVmUuid

A virtual machine uuid

sTargetHost

The name of the target host machine.

nTargetPort

The port number on the target host.

sTargetSessionId

The target Virtuozzo Service session ID.

backup_flags

Flags that specify backup type and etc. Acceptable flags:

reserved_flags

Reserved flags.

force_operation

Specifies to process VM backup action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

sDescription

The backup description.

PBT_FULL

full backup

PBT_INCREMENTAL

incremental backup

PBT_DIFFERENTIAL
differental backup

PBT_UNCOMPRESSED
uncompressed backup (by default fast zip compression used) PVMSL_LOW_SECURITY, PVMSL_NORMAL_SECURITY, PVMSL_HIGH_SECURITY

PBT_DIRECT_DATA_CONNECTION
Disables tunneling for backup (a laissez-fair data transfer).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Notes

errors are the same as for VmMigrate activity

PrlSrv_DeleteOfflineService Function

Removes an existing offline service configuration.

Syntax

```
PRL_HANDLE PrlSrv_DeleteOfflineService(
    PRL_HANDLE hServer,
    PRL_HANDLE hOffmgmtService,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer
A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hOffmgmtService
A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying a offline management service

nFlags
Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_DeleteOfflineService

PrlSrv_GetOfflineServicesList (p. 252)

PrlSrv_DeleteVirtualNetwork Function

Removes an existing virtual network from the Virtuozzo Service configuration.

Syntax

```
PRL_HANDLE PrlSrv_DeleteVirtualNetwork(
    PRL_HANDLE hServer,
    PRL_HANDLE hVirtNet,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying a virtual network.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_AddVirtualNetwork (p. 224)

PrlSrv_GetVirtualNetworkList (p. 263)

PrlSrv_UpdateVirtualNetwork (p. 312)

PrlSrv_DisableConfirmationMode Function

Disable administrator confirmation mode for the session.

Syntax

```
PRL_HANDLE PrlSrv_DisableConfirmationMode(
    PRL_HANDLE hServer,
    PRL_CONST_STR sUser,
    PRL_CONST_STR sPasswd,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

sUser

User name from administrator group (a UTF-8 encoded, null-terminated string).

sPasswd

Password of user (a UTF-8 encoded, null-terminated string).

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_DisableConfirmationMode() PrlSrv_EnableConfirmationMode (p. 236)() PrlSrv_IsConfirmationModeEnabled (p. 268)() PrlVmCfg_GetConfirmationsList (p. 561)() PrlVmCfg_SetConfirmationsList (p. 660)() PrlDispCfg_GetConfirmationsList (p. 39)() PrlDispCfg_SetConfirmationsList (p. 60)()

PrlSrv_EnableConfirmationMode Function

Enable administrator confirmation mode for the session.

Syntax

```
PRL_HANDLE PrlSrv_EnableConfirmationMode(
    PRL_HANDLE hServer,
```

```
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_DisableConfirmationMode (p. 236)() PrlSrv_EnableConfirmationMode() PrlSrv_IsConfirmationModeEnabled (p. 268)() PrlVmCfg_GetConfirmationsList (p. 561)() PrlVmCfg_SetConfirmationsList (p. 660)() PrlDispCfg_GetConfirmationsList (p. 39)() PrlDispCfg_SetConfirmationsList (p. 60)()

PrlSrv_FsCanCreateFile Function

Determines if the current user has sufficient rights to create a file or a directory in the specified location on the host.

Syntax

```
PRL_HANDLE PrlSrv_FsCanCreateFile(
    PRL_HANDLE hServer,
    PRL_CONST_STR path
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

path

Full path to the intended file or directory.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid server handle was passed or invalid path was specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_FsCreateDir Function

Creates a directory in the specified location on the host.

Syntax

```
PRL_HANDLE PrlSrv_FsCreateDir(
    PRL_HANDLE hServer,
    PRL_CONST_STR path
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

path

The new directory name and path.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

If you are creating a series of directories that relate to each other, you can use the PrlSrv_FsGenerateEntryName (p. 238) function to automatically generate their names.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid server handle was passed or invalid path was specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_FsGenerateEntryName Function

Automatically generates a unique name for a new directory.

Syntax

```
PRL_HANDLE PrlSrv_FsGenerateEntryName(
    PRL_HANDLE hServer,
    PRL_CONST_STR sDirPath,
    PRL_CONST_STR sFilenamePrefix,
    PRL_CONST_STR sFilenameSuffix,
    PRL_CONST_STR sIndexDelimiter
```

```
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sDirPath

A full path to the target location (directory).

sFilenamePrefix

Specifies a prefix to use in the directory name. Pass null or empty string to use the default prefix: tmpfile. The name will be generated by adding a number to the prefix (e.g. tmpfile1, tmpfile2, tmpfile3, etc.)

sFilenameSuffix

Specifies a suffix to use in the directory name. Pass null or an empty string value if you don't want to use a suffix.

sIndexDelimiter

A character(s) that will be inserted between the prefix and the index (e.g. tmpfile_1, tmpfile_2, tmpfile_3, etc). You can specify a null pointer or an empty string if you don't want to insert any extra characters there.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function allows to generate directory names using a specified prefix and optionally a suffix. You can specify your own prefix or use the default one. The function will not create the directories on the host. To create the directories, use the PrlSrv_FsCreateDir (p. 238) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid server handle was passed or invalid path was specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamAsString (p. 213) function to obtain the file system entry name from the PHT_RESULT (p. 210) object. The directory name is returned as a UTF-8 encoded, null-terminated string.

PrlSrv_FsGetDirEntries Function

Retrieves information about the specified file system entry residing on the specified host computer.

Syntax

```
PRL_HANDLE PrlSrv_FsGetDirEntries(
    PRL_HANDLE hServer,
    PRL_CONST_STR path
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

path

An absolute directory name and path, or drive letter (e.g. C:\) on the host computer for which to get the information.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The information retrieved by the PrlSrv_FsGetDirEntries function includes the information about the specified entry and, if the entry is a directory, the information about its immediate child entries (files and subdirectories). On Windows, you can also specify a drive letter, in which case the information about the root directory entries will be retrieved.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid server handle was passed or invalid path was specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206) containing the file system entry information.

PrlSrv_FsGetDiskList Function

Returns a list of root directories on the host computer.

Syntax

```
PRL_HANDLE PrlSrv_FsGetDiskList(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

On Windows, the function returns a list of PHT_REMOTE_FILESYSTEM_INFO (p. 206) objects containing the disk volume

information (e.g. C:\, D:\, etc.). On other operating systems, it returns a single object containing the root directory.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid server handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamsCount (p. 216) to get the number of results.
3. Use the PrlResult_GetParamByIndex (p. 213) function to get individual results.

Example

The following example illustrates how to obtain a handle of type PHT_REMOTE_FILESYSTEM_INFO (p. 206).

```
hJob =
PrlSrv_FsGetDiskList( hServer );
PrlJob_Wait( hJob, JOB_TIMEOUT );
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_REMOTE_FILESYSTEM_INFO handle.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_FsRemoveEntry Function

Removes a file system entry from the host computer.

Syntax

```
PRL_HANDLE PrlSrv_FsRemoveEntry(
    PRL_HANDLE hServer,
    PRL_CONST_STR path
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

path

An absolute path to the file system entry to be removed.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_FsRenameEntry Function

Renames a file system entry on the host.

Syntax

```
PRL_HANDLE PrlSrv_FsRenameEntry(
    PRL_HANDLE hServer,
    PRL_CONST_STR oldPath,
    PRL_CONST_STR newPath
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

oldPath

An absolute path to the entry to be renamed.

newPath

An absolute path including the new file system entry name.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_GetBackupTree Function

Get backups tree from backup server.

Syntax

```
PRL_HANDLE PrlSrv_GetBackupTree(
    PRL_HANDLE hSourceServer,
    PRL_CONST_STR sUuid,
    PRL_CONST_STR sTargetHost,
    PRL_UINT32 nTargetPort,
    PRL_CONST_STR sTargetSessionId,
    PRL_UINT32 backup_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
```

```
);
```

File

PrlApiVm.h

Parameters

hSourceServer

A handle of type PHT_SERVER (p. 219) identifying the source Virtuozzo Service.

sUuid

A virtual machine or a backup ID. If it is empty, the tree will be built for backups of all virtual machines. When PBT_BACKUP_ID or PBT_CHAIN flag is specified in the backup_flags argument, then the value of this argument is treated as a backup ID, otherwise - as a virtual machine UUID.

sTargetHost

The name of the target host machine.

nTargetPort

The port number on the target host.

sTargetSessionId

The target Virtuozzo Service session ID.

backup_flags

Flags that specify backup type and etc. Acceptable flags:

reserved_flags

Reserved flags.

force_operation

Specifies to process VM backup action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

PBT_VM

include only virtual machine backups (used by default if PBT_VM and PBT_CT flags are not specified)

PBT_CT

include only container backups

PBT_BACKUP_ID

get information about the backup, which identifier is specified in sUuid parameter. Note that if an identifier of an incremental backup is passed in the sUuid parameter, then the output tree would also contain the corresponding full backup element.

PBT_CHAIN

get information about the whole backup chain, which is dependent on the backup identifier, specified in sUuid parameter. Note that if an identifier of an incremental backup is passed in the sUuid parameter, then the output tree would also contain the corresponding full backup element.

PVMSL_LOW_SECURITY, PVMSL_NORMAL_SECURITY, PVMSL_HIGH_SECURITY

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Some of the important XML elements are:

<VmItem> - root element of a branch containing information about a virtual machine. This element, in turn, may contain a set of

elements describing full backups of this virtual machine.

<BackupItem> - root element of a branch containing information about a full backup. This element, in turn, may contain a set of elements describing incremental backups of this full backup.

<PartialBackupItem> - root element of a branch containing information about an incremental backup.

<Id> - the backup ID. The ID is required when performing other operations on a backup.

<DateTime> - the backup creation date and time.

<Type> - backup type {'f' - full, 'i' - incremental}

<Size> Backup size in bytes

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_INVALID_PARAM (p. 1293) - invalid combination of backup_flags was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get a backup tree from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamAsString (p. 213) function to obtain a string value containing the backup tree XML data.

Please note that no parser is provided to traverse the returned XML. To use this functionality, you will have to parse the returned XML yourself.

PrlSrv_GetCommonPrefs Function

Returns a handle of type PHT_DISP_CONFIG (p. 32) containing the specified Virtuozzo Service preferences information.

Syntax

```
PRL_HANDLE PrlSrv_GetCommonPrefs(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_DISP_CONFIG (p. 32) containing the Virtuozzo Service preferences.

Example

The following example illustrates how to obtain a handle of type PHT_DISP_CONFIG (p. 32).

```
hJob =
PrlSrv_GetCommonPrefs( hServer );
PrlJob_Wait( hJob, 10000 );
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_DISP_CONFIG handle.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_GetCPUPoolsList Function

Obtains a list of handles of type PHT_CPU_POOL (p. 24) containing information about all existing cpu pools.

Syntax

```
PRL_HANDLE PrlSrv_GetCPUPoolsList(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamByIndex (p. 213) function to obtain a handle of type PHT_CPU_POOL (p. 24) containing the information about a cpu pool.

PrlSrv_GetCtTemplateList Function

Retrieves a list of Containers templates registered with the specified Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_GetCtTemplateList(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The information about each template is returned as a handle of type PHT_CT_TEMPLATE (p. 26).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamsCount (p. 216) function to get the number of templates in the result set.
3. Use the PrlResult_GetParamByIndex (p. 213) function in a loop, or by passing a specific index, to obtain a handle to each template.

PrlSrv_GetDefaultVmConfig Function

Obtains a new handle of type PHT_VIRTUAL_MACHINE (p. 442).

Syntax

```
PRL_HANDLE PrlSrv_GetDefaultVmConfig(
    PRL_HANDLE hServer,
    PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR pParam,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

pParam

Pointer to the PRL_GET_VM_CONFIG_PARAM_DATA structure.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Obtaining a new PHT_VIRTUAL_MACHINE (p. 442) handle is the first step when creating a virtual machine. The second step is setting the virtual machine configuration parameters. The final step is calling the PrlVm_Reg (p. 483) function, which will create a new virtual machine.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_VIRTUAL_MACHINE (p. 442).

PrlSrv_GetDiskFreeSpace Function

Get available disk space for path.

Syntax

```
PRL_HANDLE PrlSrv_GetDiskFreeSpace(
    PRL_HANDLE hServer,
    PRL_CONST_STR sPath,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sPath

Path as pointer of type PRL_CONST_STR_PTR (p. 787).

nFlags

Flags

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamAsString (p. 213) function to obtain a result as string

PrlSrv_GetIPPrivateNetworksList Function

Obtains a list of handles of type PHT_IPPRIV_NET (p. 133) containing information about all existing private networks.

Syntax

```
PRL_HANDLE PrlSrv_GetIPPrivateNetworksList(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamByIndex (p. 213) function to obtain a handle of type PHT_IPPRIV_NET (p. 133) containing the information about a private network.

PrlSrv_GetLicenseInfo Function

Returns an object of type PHT_LICENSE (p. 146) containing the Virtuozzo license information.

Syntax

```
PRL_HANDLE PrlSrv_GetLicenseInfo(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_LICENSE (p. 146) containing the license information.

Example

The following example illustrates how to obtain a handle of type PHT_LICENSE (p. 146).

```
hJob =
PrlSrv_GetLicenseInfo( hServer );
PrlJob_Wait( hJob, 10000 );
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_LICENSE handle.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_GetNetServiceStatus Function

Returns an object of type PHT_NET_SERVICE_STATUS (p. 153) containing the Virtuozzo network service status information.

Syntax

```
PRL_HANDLE PrlSrv_GetNetServiceStatus(
    PRL_HANDLE hServer
```

```
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_NET_SERVICE_STATUS (p. 153) containing the NET service status.

Example

The following example illustrates how to obtain a handle of type PHT_NET_SERVICE_STATUS (p. 153) containing the Virtuozzo Service network service status.

```
hJob =
PrlSrv_GetNetServiceStatus( hServer );
PrlJob_Wait( hJob, 10000 );
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_NET_SERVICE_STATUS handle.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_GetNetworkClassesList Function

Obtains a list of handles of type PHT_HANDLES_LIST (p. 108) containing the list of handles of type PHT_NETWORK_CLASS (p. 154).

Syntax

```
PRL_HANDLE PrlSrv_GetNetworkClassesList(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_HANDLES_LIST (p. 108)

PrlSrv_GetNetworkShapingConfig Function

Obtains the network shaping configuration.

Syntax

```
PRL_HANDLE PrlSrv_GetNetworkShapingConfig(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169)

PrlSrv_GetOfflineServicesList Function

Obtains a list of handles of type PHT_OFFLINE_SERVICE (p. 173) containing information about all existing services.

Syntax

```
PRL_HANDLE PrlSrv_GetOfflineServicesList(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamByIndex (p. 213) function to obtain a handle of type PHT_OFFLINE_SERVICE (p. 173) containing the information about an individual offline service.

PrlSrv_GetPackedProblemReport Function

Allows to receive a packed problem report from the Virtuozzo Service in the event of a virtual machine operation failure.

Syntax

```
PRL_HANDLE PrlSrv_GetPackedProblemReport(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service. PRL_UINT32 (p. 798) nFlags - reserved flags parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get a report from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to

the PHT_RESULT (p. 210) object.

2. Use the PrlResult_GetParamAsString (p. 213) function to obtain a

string value containing the packed report dir path.

PrlSrv_GetPerfStats Function

Obtains a handle of type PHT_EVENT (p. 72) containing the perfomance statistics.

Syntax

```
PRL_HANDLE PrlSrv_GetPerfStats(
    PRL_HANDLE hServer,
    PRL_CONST_STR sFilter
);
```

File

PrlApiStat.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sFilter

String representing statistics filter, use char '*' at the end to get subset of statistic. Could be NULL, to get all statistic. See PrlPerfCounters.h header file in order to determine published performance counters patterns list.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_EVENT (p. 72).

PrlSrv_GetPluginsList Function

Gets plugin list with plugin info entries.

Syntax

```
PRL_HANDLE PrlSrv_GetPluginsList(
    PRL_HANDLE hServer,
    PRL_CONST_STR sClassId,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sClassId

[optional] Class GUID in text format.

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_GetProblemReport Function

Allows to receive a problem report from the Virtuozzo Service in the event of a virtual machine operation failure.

Syntax

```
PRL_HANDLE PrlSrv_GetProblemReport(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get a report from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamAsString (p. 213) function to obtain a string value containing the report.

PrlSrv_GetQuestions Function

The PrlSrv_GetQuestions function allows to synchronously receive questions from a Virtuozzo Service.

Syntax

```
PRL_RESULT PrlSrv_GetQuestions(
    PRL_HANDLE hServer,
    PRL_HANDLE_PTR pQuestionsList
);
```

File

PrlApiDisp.h

Parameters

pQuestionsList

[out] A pointer to a handle of type PHT_HANDLES_LIST (p. 108) containing a list of available questions.

handle

A handle of type PHT_SERVER (p. 219).

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_SERVER (p. 219) handle or non valid (NULL) pointer to the buffer for storing result was passed. PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to complete the operation. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

It can be used as an alternative to asynchronous question processing through events. You can use this function at any time to check if a question is awaiting the client response on the Virtuozzo Service.

PrlSrv_GetRestrictionInfo Function

Gets restriction info.

Syntax

```
PRL_RESULT PrlSrv_GetRestrictionInfo(
    PRL_HANDLE hServer,
    PRL_LICENSE_RESTRICTION_KEY nRestrictionKey,
    PRL_HANDLE_PTR phEvtParam
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

nRestrictionKey

Restriction key value form PRL_LICENSE_RESTRICTION_KEY (p. 1076) enumeration.

phEvtParam

[out] A pointer to a variable that receives the handle of PHT_EVENT_PARAMETER (p. 88) type.

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_SERVER (p. 219) handle or non valid (NULL) pointer to the buffer for storing result was passed. PRL_ERR_NO_DATA (p. 1324) - no info for the given restriction or restriction has boolean type. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

See Also

PrlSrv_HasRestriction (p. 266)

PrlSrv_GetServerInfo Function

Obtains a handle of type PHT_SERVER_INFO (p. 344) containing the host computer information.

Syntax

```
PRL_RESULT PrlSrv_GetServerInfo(
    PRL_HANDLE hServer,
    PRL_HANDLE_PTR phServerInfo
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

phServerInfo

[out] A pointer to a variable that receives a handle to the server info object.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Please note that in order to obtain the complete host info, the function must be called after the login operation is completed and the results of the login operation are received. In all other situations, some of the Virtuozzo Service properties may not be populated.

PrlSrv_GetSrvConfig Function

Obtains a handle of type PHT_SERVER_CONFIG (p. 314) containing the host configuration information.

Syntax

```
PRL_HANDLE PrlSrv_GetSrvConfig(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_SERVER_CONFIG (p. 314) containing the host configuration information.

Example

The following example illustrates how to obtain a handle of type PHT_SERVER_CONFIG (p. 314) from the PHT_SERVER (p. 219) object (the example assumes that you've already obtained a server handle).

```
hJob =
PrlSrv_GetSrvConfig( hServer );
PrlJob_Wait( hJob, 10000 );
PrlJob_GetResult( hJob, &hJobResult );
```

```
PrlHandle_Free( hJob );

// Get the PHT_SERVER_CONFIG handle from the result object.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_GetStatistics Function

Obtains a handle of type PHT_SYSTEM_STATISTICS (p. 360) containing the host resource usage statistics.

Syntax

```
PRL_HANDLE PrlSrv_GetStatistics(
    PRL_HANDLE hServer
);
```

File

PrlApiStat.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_SYSTEM_STATISTICS (p. 360).

Example

The following example illustrates how to obtain a handle of type PHT_SYSTEM_STATISTICS (p. 360).

```
PRL_RESULT err = PRL_ERR_UNINITIALIZED;
PRL_RESULT rc = PRL_ERR_UNINITIALIZED;
PRL_HANDLE hJob = PRL_INVALID_HANDLE;
PRL_HANDLE hJobResult = PRL_INVALID_HANDLE;
PRL_HANDLE hServer = PRL_INVALID_HANDLE;

// Initialize API library.
err = PrlApi_Init(PARALLELS_API_VER);
if (PRL_FAILED(err))
{
    // Error handling goes here...
    return -1;
}

// Create server handle.
err = PrlSrv_Create(&hServer);
```

```
// if (err != PRL_ERR_SUCCESS), process the error here.

hJob = PrlSrv_Login(
    hServer, // Server handle
    "10.30.21.243", // Server IP address
    "jdoe", // User
    "secret", // Password
    0, // Previous session ID
    0, // Port number
    0, // Timeout
    PSL_LOW_SECURITY); // Security

// Wait for the job to complete.
err = PrlJob_Wait(hJob, 1000);

// if (err != PRL_ERR_SUCCESS), process the error here.

// Get job return code.
err = PrlJob_GetRetCode(hJob, &rc);

// if (err != PRL_ERR_SUCCESS), process the error here.
// if (rc != PRL_ERR_SUCCESS), process the error here.

// Free job handle.
PrlHandle_Free(hJob);

// Get statistics.
hJob = PrlSrv_GetStatistics(hServer);
err = PrlJob_Wait(hJob, 10000);

// if (err != PRL_ERR_SUCCESS), process the error here.

err = PrlJob_GetRetCode(hJob, &rc);

// if (err != PRL_ERR_SUCCESS), process the error here.
// if (rc != PRL_ERR_SUCCESS), process the error here.

err = PrlJob_GetResult(hJob, &hJobResult);
PrlHandle_Free(hJob);

// Get the PHT_SYSTEM_STATISTICS handle.
PRL_HANDLE hHandle = PRL_INVALID_HANDLE;
err = PrlResult_GetParam(hJobResult, &hHandle);

// if (err != PRL_ERR_SUCCESS), process the error here.

PrlHandle_Free(hJobResult);
```

PrlSrv_GetSupportedOses Function

Extracts supported guest OSes matrix from remote server (it's more proper way instead of using local synch API in case when client and server interoperate remotely).

Syntax

```
PRL_RESULT PrlSrv_GetSupportedOses(
    PRL_HANDLE hServer,
    PRL_HANDLE_PTR phOsesMatrix
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

phOsesMatrix

pointer to the buffer for storing result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_NO_DATA (p. 1324) - no supported OSes data (OSes matrix filling just after login procedure completed). Also this field can be empty in case when remote server is not supported this interface.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_GUEST_OSES_MATRIX (p. 105) which will contain complete list of supported guest OSes types, versions and defaults.

PrlSrv_GetUserInfo Function

Obtains a handle of type PHT_USER_INFO (p. 371) containing information about the specified user.

Syntax

```
PRL_HANDLE PrlSrv_GetUserInfo(
    PRL_HANDLE hServer,
    PRL_CONST_STR sUserId
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sUserId

UUID of the user to obtain the information for.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_USER_INFO (p. 371) containing the user information.

See Also

PrlSrv_GetUserInfoList (p. 261)

PrlSrv_GetUserInfoList Function

Obtains a list of handles of type PHT_USER_INFO (p. 371) containing information about all known users.

Syntax

```
PRL_HANDLE PrlSrv_GetUserInfoList(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamByIndex (p. 213) function to obtain a handle of type PHT_USER_INFO (p. 371) containing an individual user information.

PrlSrv_GetUserProfile Function

Obtains a handle of type PHT_USER_PROFILE (p. 374) containing the profile data of the currently logged in user.

Syntax

```
PRL_HANDLE PrlSrv_GetUserProfile(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_USER_PROFILE (p. 374).

Example

The following example shows how to obtain a handle of type PHT_USER_PROFILE (p. 374).

```
hJob =
PrlSrv_GetUserProfile( hServer );
PrlJob_Wait( hJob, 10000 );
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_USER_PROFILE handle.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_GetVcmmdConfig Function

Obtain a handle to vcmmd config.

Syntax

```
PRL_HANDLE PrlSrv_GetVcmmdConfig(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
```

```
);
```

File

PrlApiDisp.h

Remarks

- Parameters
- hServer : A handle of type PHT_SERVER (p. 219).
- nFlags : A flags to select mode(from enum PRL_VCMMD_COMMAND_FLAGS (p. 1099))
- Returns
- A handle of type PHT_JOB (p. 138) containing the results of this
- asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not
- enough memory to instantiate the job object.

PrlSrv_GetVirtualNetworkList Function

Obtains a list of handles of type PHT_VIRTUAL_NETWORK (p. 507) containing information about all existing virtual networks.

Syntax

```
PRL_HANDLE PrlSrv_GetVirtualNetworkList(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamByIndex (p. 213) function to obtain a handle of type PHT_VIRTUAL_NETWORK (p. 507) containing the information about a virtual network.

PrlSrv_GetVmConfig Function

Get VM config by given UUID or name.

Syntax

```
PRL_HANDLE PrlSrv_GetVmConfig(
    PRL_HANDLE hServer,
    PRL_CONST_STR sSearchId,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

sSearchId

VM UUID or name.

nFlags

Search by UUID if nFlags is 0 or PGVC_SEARCH_BY_UUID, by name, if nFlags is PGVC_SEARCH_BY_NAME and search first by uuid, then by name if nFlags is PGVC_SEARCH_BY_UUID | PGVC_SEARCH_BY_NAME. PGVC_FILL_AUTOGENERATED indicates that config should include implicit autogenerated params.

hHandle

A handle of type PHT_SERVER (p. 219).

Returns

A handle of type PHT_JOB (p. 138) containing the results of operation.

PrlSrv_GetVmList Function

Retrieves a list of virtual machines registered with the specified Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_GetVmList(
    PRL_HANDLE hServer
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The information about each virtual machine is returned as a handle of type PHT_VIRTUAL_MACHINE (p. 442).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamsCount (p. 216) function to get the number of virtual machine in the result set.
3. Use the PrlResult_GetParamByIndex (p. 213) function in a loop, or by passing a specific index, to obtain a handle to each virtual machine.

Example

The following example illustrates how to obtain a list of virtual machine registered on the specified Virtuozzo Service. The examples assumes that we've already obtain a handle of type PHT_SERVER (p. 219) (the hServer variable) identifying the Virtuozzo Service.

```
hJob =
PrlSrv_GetVmList( hServer );
PrlJob_Wait( hJob, 10000 );

PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

PRL_UINT32 nIndex, nCount;

// Get the number of the virtual machines
// in the list.
PrlResult_GetParamsCount( hJobResult, &nCount );

// Iterate through the list.
for( nIndex = 0; nIndex < nCount ; nIndex++ )
{
    PRL_HANDLE hParam;
    PrlResult_GetParamByIndex( hJobResult, nIndex, &hParam );

    PRL_CHAR sBuf[1024];
    PRL_UINT32 nBufSize = sizeof( sBuf );

    // Get virtual machine name.
    nRetCode =
    PrlVm_GetName( hParam, sBuf, &nBufSize );

    if ( PRL_SUCCEEDED( nRetCode ))
        printf( "VM name: %s.\n", sBuf );
    else
        fprintf( stderr, "PrlFoundVmInfo_GetName failed, error: %s. \n",
            PRL_RESULT_TO_STRING( nRetCode ));

    PrlHandle_Free( hParam );
}
PrlHandle_Free( hJobResult );
```

PrlSrv_GetVmListEx Function

Retrieves a list of virtual machines registered with the specified Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_GetVmListEx(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Flags bitset of PRL_GET_VM_LIST_FLAGS (p. 1062) enumerator.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The information about each virtual machine is returned as a handle of type PHT_VIRTUAL_MACHINE (p. 442).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamsCount (p. 216) function to get the number of virtual machine in the result set.
3. Use the PrlResult_GetParamByIndex (p. 213) function in a loop, or by passing a specific index, to obtain a handle to each virtual machine.

PrlSrv_HasRestriction Function

Determines having restriction.

Syntax

```
PRL_RESULT PrlSrv_HasRestriction(
    PRL_HANDLE hServer,
    PRL_LICENSE_RESTRICTION_KEY nRestrictionKey,
    PRL_BOOL_PTR pbHasRestriction
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

nRestrictionKey

Restriction key value form PRL_LICENSE_RESTRICTION_KEY (p. 1076) enumeration.

pbHasRestriction

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the current license has the given restriction.

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_SERVER (p. 219) handle or non valid (NULL) pointer to the buffer for storing result was passed. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

See Also

PrlSrv_GetRestrictionInfo (p. 256)

PrlSrv_InstallAppliance Function

Sends an appliance configuration to the Virtuozzo Service in order to install the corresponding appliance from Virtuozzo download service.

Syntax

```
PRL_HANDLE PrlSrv_InstallAppliance(
    PRL_HANDLE hServer,
    PRL_HANDLE hAppCfg,
    PRL_CONST_STR sVmParentPath,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hAppCfg

A handle of type PHT_APPLIANCE_CONFIG identifying the appliance.

sVmParentPath

Optional. Name and path of the parent directory where the virtual machine appliance should be installed. If this parameter is omitted (null pointer or empty string is passed), the new virtual machine appliance directory will be created in the default directory for this Virtuozzo Service.

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627)

if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_IsConfirmationModeEnabled Function

Determines confirmation mode for the session

Syntax

```
PRL_RESULT PrlSrv_IsConfirmationModeEnabled(
    PRL_HANDLE hServer,
    PRL_BOOL_PTR pbConfirmationModeEnabled
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

pbConfirmationModeEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that confirmation mode is enabled for the session. PRL_FALSE (p. 1626) indicates that confirmation mode is disabled for the session.

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_SERVER (p. 219) handle or non valid (NULL) pointer to the buffer for storing result was passed. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

See Also

PrlSrv_DisableConfirmationMode (p. 236)() PrlSrv_EnableConfirmationMode (p. 236)() PrlSrv_IsConfirmationModeEnabled() PrlVmCfg_GetConfirmationsList (p. 561)() PrlVmCfg_SetConfirmationsList (p. 660)() PrlDispCfg_GetConfirmationsList (p. 39)() PrlDispCfg_SetConfirmationsList (p. 60)()

PrlSrv_IsConnected Function

Determines if the connection to the specified Virtuozzo Service is active.

Syntax

```
PRL_RESULT PrlSrv_IsConnected(
    PRL_HANDLE hServer,
    PRL_BOOL_PTR pbConnected,
    PRL_BOOL_PTR pbIsConnectionLocal
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

pbConnected

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the connection is active. PRL_FALSE (p. 1626) indicates otherwise.

pbIsConnectionLocal

[out] Specifies whether this is a local or remote connection. PRL_TRUE (p. 1652) indicates that this is a local connection. PRL_FALSE (p. 1626) indicates that this is a remote connection.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_IsFeatureSupported Function

Checks whether specified feature supported by remote side.

Syntax

```
PRL_RESULT PrlSrv_IsFeatureSupported(
    PRL_HANDLE hServer,
    PRL_FEATURES_MATRIX nFeatureId,
    PRL_BOOL_PTR pbIsSupported
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

nFeatureId

Requesting feature identifier

pbIsSupported

pointer to the buffer to retrieve result

Returns

PRL_ERR_INVALID_ARG (p. 1290) - either invalid/wrong handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Please note that actual data presents just after login to server operation was completed.

PrlSrv_IsNonInteractiveSession Function

Determines if the session is noninteractive or interactive.

Syntax

```
PRL_RESULT PrlSrv_IsNonInteractiveSession(
    PRL_HANDLE hServer,
    PRL_BOOL_PTR pbNonInteractive
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

pbNonInteractive

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the session is noninteractive. PRL_FALSE (p. 1626) indicates interactive session (default).

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_SERVER (p. 219) handle or non valid (NULL) pointer to the buffer for storing result was passed. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

PrlSrv_IsProxyConnection Function

Determines if the connection to the specified Virtuozzo Service is a proxy connection.

Syntax

```
PRL_RESULT PrlSrv_IsProxyConnection(
    PRL_HANDLE hServer,
    PRL_BOOL_PTR pbProxyConnection
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

pbProxyConnection

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the connection is proxy. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_JoinCPUPool Function

Join server to the default cpu pool

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.

Syntax

```
PRL_HANDLE PrlSrv_JoinCPUPool(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hCpuPool

A handle of type PHT_CPU_POOL (p. 24), defines the pool where server should be moved.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_LeaveCPUPool Function

Leaves server from any cpu pool

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.

Syntax

```
PRL_HANDLE PrlSrv_LeaveCPUPool(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_Login Function

Performs a remote login operation using the specified parameters.

Syntax

```
PRL_HANDLE PrlSrv_Login(
    PRL_HANDLE hServer,
    PRL_CONST_STR host,
    PRL_CONST_STR user,
    PRL_CONST_STR passwd,
    PRL_CONST_STR sPrevSessionUuid,
    PRL_UINT32 port_cmd,
    PRL_UINT32 timeout,
    PRL_SECURITY_LEVEL security_level
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

host

The name or IP address of the host machine (a UTF-8 encoded, null-terminated string).

user

User name (a UTF-8 encoded, null-terminated string).

passwd

Password (a UTF-8 encoded, null-terminated string).

sPrevSessionUuid

Previous session ID. This is an optional parameter that can be used in recovering from a lost Virtuozzo Service connection. The ID will be used to restore references to the tasks that were initiated in the previous session and are still running inside the Virtuozzo Service. You can pass a null or an empty string value if you are not restoring any references. See PrlSrv_AttachToLostTask (p. 224) for more information.

port_cmd

Port number. Pass 0 (zero) to use the default port.

timeout

Timeout value in milliseconds. The operation will be automatically interrupted if a connection is not established within this timeframe. Specify 0 (zero) for infinite timeout.

security_level

Communication security level to use for this connection. The security level set here will be used for all communications with the Virtuozzo Service for the duration of this session. The minimum allowable security level can be determined using the PrlDispCfg_GetMinSecurityLevel (p. 46) function.

flags

A bitset of flags. Allowed next values:

0 - no flags. It's behaviour by default (as PrlSrv_Login())

PACF_NON_INTERACTIVE_MODE

to use non-interactive mode. In interactive mode, a client may receive questions from the Virtuozzo Service, which it is expected to answer in order for the operation to continue. In non-interactive mode, the Virtuozzo Service will make decisions on its own.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_LOGIN_RESPONSE (p. 149), containing an additional Virtuozzo Service information.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

A remote login operation logs the user to a Virtuozzo Service running on a remote host. You can also use this function to log to a local host or you can use the PrlSrv_LoginLocal (p. 276) function, which is a simplified version created specifically to perform local host logins.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_LOGIN_RESPONSE (p. 149).

See Also

PrlDispCfg_GetMinSecurityLevel (p. 46)

Example

The following is a complete sample program that illustrates how to log in to a remote Virtuozzo Service.

```
#include "Parallels.h"
#include "Wrappers/SdkWrap/SdkWrap.h"
#include <stdio.h>

#ifdef _WIN_
#include <windows.h>
#else
#include <unistd.h>
#endif

const char *szVmName = "Windows XP";
const char *szServer = "127.0.0.1";
const char *szUsername = "SomeUserName";
const char *szPassword = "SomePassword";

int main( int argc, char* argv[] )
{
    PRL_HANDLE hServer;
    PRL_RESULT ret;
```

```
// Use the correct dynamic library depending on the platform.
#ifdef _WIN_
#define SDK_LIB_NAME "prl_sdk.dll"
#elif defined(_LIN_)
#define SDK_LIB_NAME "libprl_sdk.so"
#elif defined(_MAC_)
#define SDK_LIB_NAME "libprl_sdk.dylib"
#endif

// Load the SDK library.
if ( PRL_FAILED( SdkWrap_Load( SDK_LIB_NAME ) ) &&
    PRL_FAILED( SdkWrap_Load( "./" SDK_LIB_NAME ) ) )
{
    fprintf( stderr, "Failed to load " SDK_LIB_NAME "\n" );
    return -1;
}

// Initialize the API.
ret = PrlApi_Init( Virtuozzo _API_VER );
if ( PRL_FAILED( ret ) )
{
    fprintf( stderr, "PrlApi_Init returned with error: %s.\n",
        PRL_RESULT_TO_STRING( ret ) );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return ret;
}

// Create a server handle.
ret = PrlSrv_Create( &hServer );
if ( PRL_FAILED( ret ) )
{
    fprintf( stderr, "PrlSvr_Create failed, error: %s",
        PRL_RESULT_TO_STRING( ret ) );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Log in (asynchronous call).
PRL_HANDLE hJob = PrlSrv_Login(
        hServer,            // PRL_HANDLE of type PHT_SERVER.
        szServer,           // Name and IP address of the host machine.
        szUsername,         // User name.
        szPassword,         // Password.
        0,                  // Previuos session ID (0 if not used).
        0,                  // Port number (0 for default port).
        0,                  // Timeout value (0 for default timeout).
        PSL_LOW_SECURITY ); // Security level.

// Wait for a maximum of 10 seconds for
// the job to complete.
ret = PrlJob_Wait( hJob, 1000 );
if ( PRL_FAILED( ret ) )
{
    fprintf( stderr, "PrlJob_Wait for PrlSrv_Login returned with error: %s\n",
        PRL_RESULT_TO_STRING( ret) );
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}
```

```
// Analyze the result of PrlSrv_Login.
PRL_RESULT nJobResult;
ret = PrlJob_GetRetCode( hJob, &nJobResult );
if ( PRL_FAILED( nJobResult ) )
{
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    printf( "Login job returned with error: %s\n",
        PRL_RESULT_TO_STRING( nJobResult ) );
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}
else
    printf( "Login was successful.\n" );

// Log off.
hJob = PrlSrv_Logoff( hServer );
ret = PrlJob_Wait( hJob, 1000 );
if ( PRL_FAILED( ret ) )
{
    fprintf( stderr, "PrlJob_Wait for PrlSrv_Logoff returned error: %s\n",
        PRL_RESULT_TO_STRING( ret ) );
    PrlHandle_Free( hVm );
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

ret = PrlJob_GetRetCode( hJob, &nJobResult );
if ( PRL_FAILED( ret ) )
{
    fprintf( stderr, "PrlJob_GetRetCode failed for PrlSrv_Logoff with error: %s\n",
        PRL_RESULT_TO_STRING( ret ) );
    PrlHandle_Free( hVm );
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Report success or failure of PrlSrv_Logoff.
if ( PRL_FAILED( nJobResult ) )
{
    fprintf( stderr, "PrlSrv_Logoff failed with error: %s\n",
        PRL_RESULT_TO_STRING( nJobResult ) );
    PrlHandle_Free( hVm );
    PrlHandle_Free( hJob );
    PrlHandle_Free( hServer );
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}
else
    printf( "Logoff was successful.\n" );

// Free handles that are no longer required.
PrlHandle_Free( hVm );
PrlHandle_Free( hJob );
PrlHandle_Free( hServer );
```

```
    // De-initialize the Virtuozzo API, and unload the SDK.
    PrlApi_Deinit();
    SdkWrap_Unload();

    printf( "\n\nEnd of program.\n\n" );
    char c;
    scanf( &c, "%c" );

    return 0;
}
```

PrlSrv_LoginEx Function

Syntax

```
PRL_HANDLE PrlSrv_LoginEx(
    PRL_HANDLE hServer,
    PRL_CONST_STR host,
    PRL_CONST_STR user,
    PRL_CONST_STR passwd,
    PRL_CONST_STR sPrevSessionUuid,
    PRL_UINT32 port_cmd,
    PRL_UINT32 timeout,
    PRL_SECURITY_LEVEL security_level,
    PRL_UINT32 flags
);
```

File

PrlApiDisp.h

PrlSrv_LoginLocal Function

Allows to log in to the local Virtuozzo Service (the Virtuozzo Service running on this host) using the current user credentials.

Syntax

```
PRL_HANDLE PrlSrv_LoginLocal(
    PRL_HANDLE hServer,
    PRL_CONST_STR sPrevSessionUuid,
    PRL_UINT32 port,
    PRL_SECURITY_LEVEL security_level
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sPrevSessionUuid

Previous session ID. This is an optional parameter that can be used in recovering from a lost Virtuozzo Service connection. The ID will be used to restore references to the tasks that were initiated in the previous session and are still running inside the Virtuozzo Service. You can pass a null or an empty string value if you are not restoring any references. See PrlSrv_AttachToLostTask (p. 224) for more information.

port

Port number. Specify 0 (zero) to use the default port.

security_level

Communication security level to use for this connection. The security level set here will be used for all communications with the Virtuozzo Service for the duration of this session. The minimum allowable security level can be determined using the PrlDispCfg_GetMinSecurityLevel (p. 46) function.

flags

A bitset of flags. Allowed next values:

0 - no flags. It's behaviour by default (as PrlSrv_LoginLocal())

PACF_NON_INTERACTIVE_MODE

to use non-interactive mode. In interactive mode, a client may receive questions from the Virtuozzo Service, which it is expected to answer in order for the operation to continue. In non-interactive mode, the Virtuozzo Service will make decisions on its own.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_LOGIN_RESPONSE (p. 149), containing an additional Virtuozzo Service information.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

When performing a local login operation, no password/username is needed. The authentication and authorization will be performed using the current user account.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_LOGIN_RESPONSE (p. 149).

See Also

PrlDispCfg_GetMinSecurityLevel (p. 46)

Example

The following example illustrates how to log in to locally running Virtuozzo Service and to obtain a handle of type PHT_LOGIN_RESPONSE (p. 149).

```
// Log in.
hJob = PrlSrv_LoginLocal( hServer, NULL, 0, PSL_LOW_SECURITY );
PrlJob_Wait( hJob, 10000 );

// Get the results of the login operation.
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the PHT_LOGIN_RESPONSE handle from the job result.
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlSrv_LoginLocalEx Function

Syntax

```
PRL_HANDLE PrlSrv_LoginLocalEx(
    PRL_HANDLE hServer,
    PRL_CONST_STR sPrevSessionUuid,
    PRL_UINT32 port,
    PRL_SECURITY_LEVEL security_level,
    PRL_UINT32 flags
);
```

File

PrlApiDisp.h

PrlSrv_Logoff Function

Logs off the Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_Logoff(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function logs of the Virtuozzo Service and ends a session that was started with the PrlSrv_Login (p. 272) or the PrlSrv_LoginLocal (p. 276) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_LookupParallelsServers@PRL_UINT32@PRL_EVENT_HANDLER_PTR@PRL_VOID_PTR

Example

The following example illustrates how to call the PrlSrv_LookupParallelsServers synchronously.

```
hJob =
PrlSrv_LookupParallelsServers(
    10000,  // timeout
    NULL,   // callback function (not used)
    NULL    // user object pointer (not used)
);

printf("Searching for Virtuozzo Services...\n");

nRetCode =
PrlJob_Wait(hJob, 100000);

if (PRL_FAILED(nRetCode))
{
    fprintf(stderr, "PrlJob_Wait returned with error: %s.\n",
        PRL_RESULT_TO_STRING( nRetCode));
    PrlApi_Deinit();
    SdkWrap_Unload();
    return nRetCode;
}

PrlJob_GetResult(hJob, &hJobResult);
PrlHandle_Free(hJob);

PRL_UINT32 nIndex, nCount;
PrlResult_GetParamsCount(hJobResult, &nCount);

for (nIndex = 0; nIndex < nCount ; nIndex++)
{
    PRL_HANDLE hParam;
    PrlResult_GetParamByIndex(hJobResult, nIndex, &hParam);

    PRL_CHAR sBuf[1024];
    PRL_UINT32 nBufSize = sizeof(sBuf);

    nRetCode =
    PrlSrvInfo_GetHostName(hParam, sBuf, &nBufSize);

    if (PRL_SUCCEEDED(nRetCode))
        printf("Found Service: %s.\n", &sBuf[0]);
    else
        fprintf(stderr, "PrlSrvInfo_GetHostName failed, error: %s. \n",
            PRL_RESULT_TO_STRING(nRetCode));

    PrlHandle_Free(hParam);
}

PrlHandle_Free(hJobResult);
```

In the following example, the PrlSrv_LookupParallelsServers is called asynchronously. In order to that, we first have to implement a callback function (let's call it ourCallback):

```
static PRL_RESULT ourCallback(PRL_HANDLE hEvent, PRL_VOID_PTR pUserData)
{
```

```
    printf("%s: ", pUserData);

    // Get the name of the host machine.
    PRL_UINT32 nBufSize = 1024;
    PRL_CHAR sBuf[nBufSize];
    PrlSrvInfo_GetHostName(hEvent, sBuf, &nBufSize);

    // Get the other properties and process them here...

    // The handle must be freed.
    PrlHandle_Free(hEvent);
    return rc;
}
```

The PrlSrv_LookupParallelsServers function can now be called as follows:

```
hJob =
PrlSrv_LookupParallelsServers(
    1000,
    &ourCallback,
    (PRL_VOID_PTR)("callback")
);
```

PrlSrv_MoveToCPUPool Function

Moves server to a different cpu pool

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.

Syntax

```
PRL_HANDLE PrlSrv_MoveToCPUPool(
    PRL_HANDLE hServer,
    PRL_CONST_STR sCpuPool
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sCpuPool

name of the pool where server should be moved.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_NetServiceRestart Function

Restarts the Virtuozzo network service.

Syntax

```
PRL_HANDLE PrlSrv_NetServiceRestart(
    PRL_HANDLE hServer
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_NetServiceRestoreDefaults Function

Restores the default settings of the Virtuozzo network service.

Syntax

```
PRL_HANDLE PrlSrv_NetServiceRestoreDefaults(
    PRL_HANDLE hServer
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_NetServiceStart Function

Starts the Virtuozzo network service.

Syntax

```
PRL_HANDLE PrlSrv_NetServiceStart(
    PRL_HANDLE hServer
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_NetServiceStop Function

Stops the Virtuozzo network service.

Syntax

```
PRL_HANDLE PrlSrv_NetServiceStop(
    PRL_HANDLE hServer
);
```

File

PrlApiNet.h

Parameters

unnamed_param

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_RecalculateCPUPool Function

Recalculates features in pool

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function.

Syntax

```
PRL_HANDLE PrlSrv_RecalculateCPUPool(
    PRL_HANDLE hServer,
    PRL_CONST_STR sCpuPool
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sCpuPool

name of the pool where features should be recalculated.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_RefreshPlugins Function

Refreshes states of installed Virtuozzo plugins.

Syntax

```
PRL_HANDLE PrlSrv_RefreshPlugins(
    PRL_HANDLE hServer,
```

```
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_RegEventHandler Function

Registers an event handler.

Syntax

```
PRL_RESULT PrlSrv_RegEventHandler(
    PRL_HANDLE hServer,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR userData
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

handler

A pointer to a user callback function (event handler).

userData

Optional. A pointer to a user data that will be passed to the callback function.

Remarks

Asynchronous functions return data to the caller by means of a callback function (or event handler). The callback function could be called at any time, depending on how long the asynchronous function takes to complete. The callback function must have a specific signature. The callback function prototype type is PRL_EVENT_HANDLER_PTR (p. 780). The PrlSrv_RegEventHandler function is used to register an event handler with the specified Virtuozzo Service, PrlSrv_UnregEventHandler (p. 306) is used to unregister an event handler. When an event handler is registered, it will start

receiving all events/jobs generated by the Virtuozzo Service. It is the responsibility of the client application to identify and handle the relevant event(s). For more information on events and asynchronous functions, see the Virtuozzo Virtualization SDK Programmer's Guide.

PrlSrv_Register3rdPartyVm Function

Registers an existing 3rd party virtual machine with Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_Register3rdPartyVm(
    PRL_HANDLE hServer,
    PRL_CONST_STR strVmConfigPath,
    PRL_CONST_STR strVmRootDirPath,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

strVmConfigPath

[required] An absolute path to the virtual machine configuration file (*.vmx for example)

strVmRootDirPath

[optional] An absolute path to the virtual machine root directory where virtual machine home directory will be created.

nFlags

Flags bitset: 0 - Default behavior PR3F_ALLOW_UNKNOWN_OS

- to allow undetectable OS

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_VIRTUAL_MACHINE (p. 442) containing information about the virtual machine that was registered.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_VIRTUAL_MACHINE (p. 442).

PrlSrv_RegisterVm Function

Registers an existing virtual machine with Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_RegisterVm(
    PRL_HANDLE hServer,
    PRL_CONST_STR strVmDirPath,
    PRL_BOOL bNonInteractiveMode
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

strVmDirPath

An absolute path to the virtual machine directory to register.

bNonInteractiveMode

Set to PRL_TRUE (p. 1652) to use non-interactive mode. Set to PRL_FALSE (p. 1626) to use interactive mode. In interactive mode, a client may receive questions from the Virtuozzo Service, which it is expected to answer in order for the operation to continue. In non-interactive mode, the Virtuozzo Service will make decisions on its own.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_VIRTUAL_MACHINE (p. 442) containing information about the virtual machine that was registered.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_VIRTUAL_MACHINE (p. 442).

Example

The following sample function demonstrates how to add an existing virtual machine to the Virtuozzo Service. The function takes a handle of type PHT_SERVER (p. 219) and a string specifying the name and path of the virtual machine directory. It registers the virtual machine with the Service and then modifies the MAC address of every virtual network adapter installed in it.

```
PRL_RESULT RegisterExistingVM(PRL_HANDLE hServer, PRL_CONST_STR sVmDirectory)
{
    PRL_HANDLE hJob = PRL_INVALID_HANDLE;
    PRL_HANDLE hJobResult = PRL_INVALID_HANDLE;

    PRL_RESULT ret = PRL_ERR_UNINITIALIZED;
    PRL_RESULT nJobRetCode = PRL_ERR_UNINITIALIZED;
```

```
// Register the virtual machine with the Virtuozzo Service.
hJob = PrlSrv_RegisterVm(
    hServer,
    sVmDirectory,
    PRL_TRUE);  // Using non-interactive mode.

ret = PrlJob_Wait(hJob, 10000);

// Check the return code of PrlSrv_RegisterVm.
PrlJob_GetRetCode(hJob, &nJobRetCode);
if (PRL_FAILED(nJobRetCode))
{
    printf("PrlSrv_RegisterVm returned error: %s\n",
        prl_result_to_string(nJobRetCode));
    PrlHandle_Free(hJob);
    return -1;
}

// Obtain the virtual machine handle from the job object.
// We will use the handle later to modify the virtual machine
// configuration.
PRL_HANDLE hVm = PRL_INVALID_HANDLE;
ret = PrlJob_GetResult(hJob, &hJobResult);
if (PRL_FAILED(ret))
{
    // Handle the error...
    return ret;
}

ret = PrlResult_GetParam(hJobResult, &hVm);
if (PRL_FAILED(ret))
{
    // Handle the error...
    return ret;
}

PrlHandle_Free(hJob);
PrlHandle_Free(hJobResult);

printf("Virtual machine '%s' was successfully registered.",
        sVmDirectory);

// The following code will generate and set a new MAC address
// for every virtual network adapter that exists in the virtual machine.
// This step is optional and should normally be performed when the virtual
// machine is a copy of another virtual machine.

PRL_HANDLE hJobBeginEdit;
PRL_HANDLE hJobCommit;

// Begin the virtual machine editing operation.
hJobBeginEdit = PrlVm_BeginEdit(hVm);
ret = PrlJob_Wait(hJobBeginEdit, 10000);
PrlJob_GetRetCode(hJobBeginEdit, &nJobRetCode);
if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Error: %s\n", prl_result_to_string(nJobRetCode));
    PrlHandle_Free(hJobBeginEdit);
    return nJobRetCode;
}

// Obtain a handle of type PHT_VM_CONFIGURATION containing the
// virtual machine configuration data.
PRL_HANDLE hVmCfg = PRL_INVALID_HANDLE;
ret = PrlVm_GetConfig(hVm, &hVmCfg);
```

```
    if (PRL_FAILED(ret))
    {
        // Handle the error...
        return ret;
    }

    // Determine the number of the network adapters
    // installed in the machine.
    PRL_UINT32 nCount = PRL_ERR_UNINITIALIZED;
    ret = PrlVmCfg_GetNetAdaptersCount(hVmCfg, &nCount);
    if (PRL_FAILED(ret))
    {
        // Handle the error...
        return ret;
    }

    // Itereate through the adapter list.
    PRL_HANDLE hNetAdapter = PRL_INVALID_HANDLE;
    for (PRL_UINT32 i = 0; i < nCount; ++i)
    {
        ret = PrlVmCfg_GetNetAdapter(hVmCfg, i, &hNetAdapter);
        if (PRL_FAILED(ret))
        {
            // Handle the error...
            return ret;
        }

        // Automatically generate new MAC address for the current adapter.
        // The address will be updated in the configuration object.
        ret = PrlVmDevNet_GenerateMacAddr(hNetAdapter);
        if (PRL_FAILED(ret))
        {
            // Handle the error...
            return ret;
        }
    }

    // Commit the changes to the virtual machine.
    hJobCommit = PrlVm_Commit(hVm);

    // Check the results of the commit operation.
    ret = PrlJob_Wait(hJobCommit, 10000);
    PrlJob_GetRetCode(hJobCommit, &nJobRetCode);
    if (PRL_FAILED(nJobRetCode))
    {
        fprintf(stderr, "Commit error: %s\n", prl_result_to_string(nJobRetCode));
        PrlHandle_Free(hJobCommit);
        return nJobRetCode;
    }

    PrlHandle_Free(hNetAdapter);
    PrlHandle_Free(hVm);
    PrlHandle_Free(hVmCfg);
    PrlHandle_Free(hJobCommit);
    PrlHandle_Free(hJobBeginEdit);

    return 0;
}
```

PrlSrv_RegisterVmEx Function

Registers an existing virtual machine with Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_RegisterVmEx(
    PRL_HANDLE hServer,
    PRL_CONST_STR strVmDirPath,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

strVmDirPath

An absolute path the virtual machine directory to register.

nFlags

A bitset of flags. Allowed next values:

0 - no flags. It's behaviour by default (as PrlSrv_RegisterVm (p. 286)())

PACF_NON_INTERACTIVE_MODE

to use non-interactive mode. In interactive mode, a client may receive questions from the Virtuozzo Service, which it is expected to answer in order for the operation to continue. In non-interactive mode, the Virtuozzo Service will make decisions on its own.

PRVF_REGENERATE_VM_UUID

to generate new VM identity after registration.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_VIRTUAL_MACHINE (p. 442) containing information about the virtual machine that was registered.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_VIRTUAL_MACHINE (p. 442).

PrlSrv_RegisterVmWithUuid Function

Registers an existing virtual machine with custom Vm UUID.

Syntax

```
PRL_HANDLE PrlSrv_RegisterVmWithUuid(
    PRL_HANDLE hServer,
    PRL_CONST_STR strVmDirPath,
```

```
    PRL_CONST_STR strVmUuid,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

strVmDirPath

An absolute path the virtual machine directory to register.

strVmUuid

Assign custom Vm uuid

nFlags

A bitset of flags. Allowed next values:

0 - no flags. It's behaviour by default (as PrlSrv_RegisterVm (p. 286)())

PACF_NON_INTERACTIVE_MODE

to use non-interactive mode. In interactive mode, a client may receive questions from the Virtuozzo Service, which it is expected to answer in order for the operation to continue. In non-interactive mode, the Virtuozzo Service will make decisions on its own.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_VIRTUAL_MACHINE (p. 442) containing information about the virtual machine that was registered.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_VIRTUAL_MACHINE (p. 442).

PrlSrv_RemoveCtTemplate Function

Removes the specified Containers template.

Syntax

```
PRL_HANDLE PrlSrv_RemoveCtTemplate(
    PRL_HANDLE hServer,
    PRL_CONST_STR sName,
    PRL_CONST_STR sOsTmplName,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sName

Name of template to be removed.

sOsTmplName

Name of OS template for which this application template is. Should be filled for application templates.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_RemoveIPPrivateNetwork Function

Removes an existing private network from the Virtuozzo Service configuration.

Syntax

```
PRL_HANDLE PrlSrv_RemoveIPPrivateNetwork(
    PRL_HANDLE hServer,
    PRL_HANDLE hPrivNet,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying a private network.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627)

if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_AddIPPrivateNetwork (p. 223)

PrlSrv_GetIPPrivateNetworkList

PrlSrv_UpdateIPPrivateNetwork (p. 308)

PrlSrv_RemoveVmBackup Function

Remove backup of the virtual machine from backup server.

Syntax

```
PRL_HANDLE PrlSrv_RemoveVmBackup(
    PRL_HANDLE hSourceServer,
    PRL_CONST_STR sVmUuid,
    PRL_CONST_STR sBackupUuid,
    PRL_CONST_STR sTargetHost,
    PRL_UINT32 nTargetPort,
    PRL_CONST_STR sTargetSessionId,
    PRL_UINT32 remove_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hSourceServer

A handle of type PHT_SERVER (p. 219) identifying the source Virtuozzo Service.

sVmUuid

A virtual machine uuid

sBackupUuid

A backup uuid

sTargetHost

The name of the target host machine.

nTargetPort

The port number on the target host.

sTargetSessionId

The target Virtuozzo Service session ID.

remove_flags

Flags that specify remove type and etc. Acceptable flags: PVMSL_LOW_SECURITY, PVMSL_NORMAL_SECURITY,

PVMSL_HIGH_SECURITY

reserved_flags

Reserved flags.

force_operation

Specifies to process VM backup action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Notes

errors are the same as for VmMigrate activity

PrlSrv_RestoreVmBackup Function

Restore backup of the virtual machine from backup server.

Syntax

```
PRL_HANDLE PrlSrv_RestoreVmBackup(
    PRL_HANDLE hSourceServer,
    PRL_CONST_STR sVmUuid,
    PRL_CONST_STR sBackupUuid,
    PRL_CONST_STR sTargetHost,
    PRL_UINT32 nTargetPort,
    PRL_CONST_STR sTargetSessionId,
    PRL_CONST_STR sTargetVmHomePath,
    PRL_CONST_STR sTargetVmName,
    PRL_UINT32 restore_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hSourceServer

A handle of type PHT_SERVER (p. 219) identifying the source Virtuozzo Service.

sVmUuid

A virtual machine uuid

sBackupUuid

A backup uuid

sTargetHost

The name of the target host machine.

nTargetPort

The port number on the target host.

sTargetSessionId

The target Virtuozzo Service session ID.

sTargetVmHomePath

Home directory of the target virtual machine.

restore_flags

Flags that specify restore type and etc. Acceptable flags:

reserved_flags

Reserved flags.

force_operation

Specifies to process VM backup action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

PBT_RESTORE_TO_COPY

change Vm name/uuid/MAC address on restore PVMSL_LOW_SECURITY, PVMSL_NORMAL_SECURITY, PVMSL_HIGH_SECURITY

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Notes

errors are the same as for VmMigrate activity

PrlSrv_SendAnswer Function

Sends an answer to the Virtuozzo Service in response to a question.

Syntax

```
PRL_HANDLE PrlSrv_SendAnswer(
    PRL_HANDLE hServer,
    PRL_HANDLE hAnswer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hAnswer

A handle of type PHT_EVENT (p. 72) containing the answer.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Some operations have an option to be carried out in interactive mode. This means that the Virtuozzo Service may interrupt an operation and ask client a question about how it would like to proceed. This function allows to send such an answer to the Virtuozzo Service. The answer must be prepared using the PrlEvent_CreateAnswerEvent (p. 74) function before it can be sent to the Virtuozzo Service using this function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_SendProblemReport Function

Requests a server to send a problem report with possibility to append also the problem report assemblied at the client's side.

Syntax

```
PRL_HANDLE PrlSrv_SendProblemReport(
    PRL_HANDLE hServer,
    PRL_HANDLE hProblemReport,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

handle of type PHT_SERVER (p. 219) of the server

hProblemReport

the report assemblied at the client's side

nFlags

flags associated with the operation (see PRL_PROBLEM_REPORT_FLAGS (p. 1086))

Returns

PRL_HANDLE (p. 820) - handle to the job generated

Remarks

The report should be created as follows: 1) with PrlApi_CreateProblemReport (p. 729) call 2) filled with PrlReport_Assembly (p. 192)

Similar calls are PrlApi_SendProblemReport (p. 742) and PrlApi_SendPackedProblemReport (p. 741), PrlReport_Send (p. 198)

Async response should normally contain report id.

PrlSrv_SetNonInteractiveSession Function

Sets the session in noninteractive or interactive mode.

Syntax

```
PRL_HANDLE PrlSrv_SetNonInteractiveSession(
    PRL_HANDLE hServer,
    PRL_BOOL bNonInteractive,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219).

bNonInteractive

Noninteractive session mode.

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_SetVcmmdConfig Function

Set handle of vcmmd config.

Syntax

```
PRL_HANDLE PrlSrv_SetVcmmdConfig(
    PRL_HANDLE hDispConfig,
    PRL_HANDLE hVcmmdConfig,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Remarks

- Parameters

- hServer : A handle of type PHT_SERVER (p. 219).
- hVcmmdConfig : A handle of type PHT_VCMMD_CONFIG (p. 379).
- nFlags : reserver parameter
- Returns
- A handle of type PHT_JOB (p. 138) containing the results of this
- asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not
- enough memory to instantiate the job object.

PrlSrv_SetVNCEncryption Function

Set VNC encryption.

Syntax

```
PRL_HANDLE PrlSrv_SetVNCEncryption(
    PRL_HANDLE hServer,
    PRL_CONST_STR sPubKey,
    PRL_CONST_STR sPrivKey,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

Parameters

sPubKey

SSL certificate to encrypt the VNC traffic

sPrivKey

SSL private key to encrypt the VNC traffic

hHandle

A handle of type PHT_SERVER (p. 219).

Returns

A handle of type PHT_JOB (p. 138) containing the results of operation.

Remarks

To delete sertificate the empty sPubKey and sPrivKey should be specified.

PrlSrv_Shutdown Function

Shuts down the Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_Shutdown(
    PRL_HANDLE hServer,
    PRL_BOOL bForceShutdown
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

bForceShutdown

Specifies whether the shutdown operation should be forced. The value of PRL_TRUE (p. 1652) will force the shutdown; PRL_FALSE (p. 1626) will perform a normal shutdown. If one or more virtual machines are running, clients are connected, or some tasks are currently in progress, then forcing the shutdown will stop all processes automatically and shut down the machine.

nFlags

A flags to select shutdown mode(from enum PRL_SRV_SHUTDOWN_FLAGS (p. 1093))

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The current user must have sufficient privileges to execute the Virtuozzo Service shutdown.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_ACCESS_TOKEN_INVALID (p. 1177) - the user doesn't have sufficient privileges.

PRL_ERR_ANOTHER_USER_SESSIONS_PRESENT (p. 1183) - another user is currently connected to the Virtuozzo Service, the Virtuozzo Service shutdown aborted.

PRL_ERR_SOME_VMS_RUNNING (p. 1356) - running virtual machines found, stop the running machines first.

PRL_ERR_SOME_TASKS_PRESENT (p. 1355) - running tasks detected, wait for the tasks to finish or cancel them.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_ShutdownEx Function

Syntax

```
PRL_HANDLE PrlSrv_ShutdownEx(
    PRL_HANDLE hServer,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

PrlSrv_StartSearchVms Function

Searches for unregistered virtual machines at the specified location(s).

Syntax

```
PRL_HANDLE PrlSrv_StartSearchVms(
    PRL_HANDLE hServer,
    PRL_HANDLE hStringsList
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hStringsList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of pathnames to search. If the list is empty, all available drives and directories will be included in search scope.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function can be used to search for virtual machines that are not currently registered with the specified Virtuozzo Service but are otherwise valid machines to be registered with it. The virtual machine information is returned as a list of PHT_FOUND_VM_INFO (p. 102) objects. When the function is executed asynchronously using callback functionality, the callback function receives the information as an event of type PET_DSP_EVT_FOUND_LOST_VM_CONFIG.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamsCount (p. 216) function to get the number of virtual machines in the list.
3. Use the PrlResult_GetParam (p. 210) function to obtain individual handles of type PHT_FOUND_VM_INFO (p. 102).

Example

```
PRL_HANDLE hStringList;
PrlApi_CreateStringsList( &hStringList );
PrlStrList_AddItem( hStringList, "/Users/Shared/Parallels/" );

hJob =
PrlSrv_StartSearchVms( hServer, hStringList );
PrlJob_Wait( hJob, 10000 );

PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

PRL_UINT32 nIndex, nCount;
PrlResult_GetParamsCount( hJobResult, &nCount );
for( nIndex = 0; nIndex < nCount ; nIndex++ )
{
    PRL_HANDLE hParam;
    PrlResult_GetParamByIndex( hJobResult, nIndex, &hParam );

    PRL_HANDLE_TYPE nHandleType;
```

```
    PrlHandle_GetType( hParam, &nHandleType );

    printf( "Handle Type: %s\n",
        PRL_HANDLE_TYPE_TO_STRING( nHandleType ));

    PRL_CHAR sBuf[1024];
    PRL_UINT32 nBufSize = sizeof( sBuf );

    nRetCode =
    PrlFoundVmInfo_GetName( hParam, sBuf, &nBufSize );

    if ( PRL_SUCCEEDED( nRetCode ))
        printf( "VM name: %s\n.", sBuf );
    else
        fprintf( stderr, "PrlFoundVmInfo_GetName failed, error: %s. \n",
            PRL_RESULT_TO_STRING( nRetCode ));

    PrlHandle_Free( hParam );
}
PrlHandle_Free( hJobResult );
PrlHandle_Free( hStringList );
```

PrlSrv_StopInstallAppliance Function

Stops the appliance installation previously started with the PrlSrv_InstallAppliance (p. 267) function call.

Syntax

```
PRL_HANDLE PrlSrv_StopInstallAppliance(
    PRL_HANDLE hServer,
    PRL_HANDLE hAppCfg,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hAppCfg

A handle of type PHT_APPLIANCE_CONFIG identifying the appliance. This must be the same handle instance that was passed to the PrlSrv_InstallAppliance (p. 267) function.

nFlags

Reserved parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_SubscribeToHostStatistics Function

Allows to subscribe to receive host OS statistics on a periodic basis.

Syntax

```
PRL_HANDLE PrlSrv_SubscribeToHostStatistics(
    PRL_HANDLE hServer
);
```

File

PrlApiStat.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To obtain statistics, an event handler (callback function) is required. Within the event handler, first check the type of event. Events of type PET_DSP_EVT_HOST_STATISTICS_UPDATED indicate an event containing statistics data. To access the statistics handle (a handle of type PHT_SYSTEM_STATISTICS (p. 360)), first extract the event parameter using PrlEvent_GetParam (p. 84), then convert the result (which will be a handle to an object of type PHT_EVENT_PARAMETER (p. 88)) to a handle using PrlEvtPrm_ToHandle (p. 92). The functions that operate on PHT_SYSTEM_STATISTICS (p. 360) references can then be used to obtain statistics data.

For the event handler to be called, it is necessary to register it with PrlSrv_RegEventHandler (p. 284). When statistics data is no longer required, unsubscribe from statistics events using PrlSrv_UnsubscribeFromHostStatistics (p. 307). When events are no longer required, unregister the event handler using PrlSrv_UnregEventHandler (p. 306).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Example

The following is a complete example that demonstrates how to obtain statistics data asynchronously using PrlSrv_SubscribeToHostStatistics. Note that the same code could be used to receive statistics data for a virtual machine, instead of the host computer, by using PrlVm_SubscribeToGuestStatistics (p. 495) instead of PrlSrv_SubscribeToHostStatistics, and passing it a handle to a virtual machine that is running. This would also require using PrlVm_UnsubscribeFromGuestStatistics (p. 501) to stop receiving statistics data for the virtual machine.

```
#include "Parallels.h"
#include "Wrappers/SdkWrap/SdkWrap.h"
#include <stdio.h>

#ifdef _WIN_
#include <windows.h>
#else
#include <unistd.h>
#endif
```

```
const char *szServer = "123.123.123.123";
const char *szUsername = "Your Username";
const char *szPassword = "Your Password";

// ------------------------------------------------------------------------
// Event handler.
// ------------------------------------------------------------------------
// 1. Check for events of type PET_DSP_EVT_HOST_STATISTICS_UPDATES.
// 2. Display a header if first call to this event handler.
// 3. Get the event param (PHT_EVENT_PARAMETER) from the PHT_EVENT handle.
// 4. Convert event param to a handle (will be type PHT_SYSTEM_STATISTICS).
// 5. Use PHT_SYSTEM_STATISTICS handle to obtain CPU usage, memory usage,
// and disk usage data.
// ------------------------------------------------------------------------
static PRL_RESULT OurCallback(PRL_HANDLE handle, void *pData)
{
    PRL_HANDLE_TYPE nHandleType;
    PRL_RESULT ret = PrlHandle_GetType(handle, &nHandleType);

    // Check for PrlHandle_GetType error here.
    if (nHandleType == PHT_EVENT)
    {
        PRL_EVENT_TYPE EventType;
        PrlEvent_GetType(handle, &EventType);

        // Check if the event type is a statistics update.
        if (EventType == PET_DSP_EVT_HOST_STATISTICS_UPDATED)
        {
            // Output a header if first call to this function.
            static PRL_BOOL bHeaderHasBeenPrinted = PRL_FALSE;
            if (!bHeaderHasBeenPrinted)
            {
                bHeaderHasBeenPrinted = PRL_TRUE;
                printf("CPU (%%) Used RAM (MB) Free RAM (MB) Used Disk Space (MB)"
                       " Free Disk Space (MB)\n");
                printf("-------------------------------------------------------"
                       "-------------------\n");
            }

            PRL_HANDLE hEventParameters;
            PRL_HANDLE hServerStatistics;

            // Get the event parameter (PHT_EVENT_PARAMETER) from the event handle.
            PrlEvent_GetParam(handle, 0, &hEventParameters);

            // Convert the event parameter to a handle (PHT_SYSTEM_STATISTICS).
            PrlEvtPrm_ToHandle(hEventParameters, &hServerStatistics);

            // Get CPU statistics (usage in %).
            PRL_HANDLE hCpuStatistics;
            ret = PrlStat_GetCpuStat(hServerStatistics, 0, &hCpuStatistics);
            PRL_UINT32 nCpuUsage = 0;
            ret = PrlStatCpu_GetCpuUsage(hCpuStatistics, &nCpuUsage);

            // Get RAM statistics.
            PRL_UINT64 nUsedRam, nFreeRam;
            PrlStat_GetFreeRamSize(hServerStatistics, &nFreeRam);
            PrlStat_GetUsageRamSize(hServerStatistics, &nUsedRam);
            nUsedRam /= (1024 * 1024);
            nFreeRam /= (1024 * 1024);

            // Get disk space statistics.
            PRL_UINT64 nFreeDiskSpace, nUsedDiskSpace;
            PRL_HANDLE hDiskStatistics;
            PrlStat_GetDiskStat(hServerStatistics, 0, &hDiskStatistics);
```

```
            PrlStatDisk_GetFreeDiskSpace(hDiskStatistics, &nFreeDiskSpace);
            PrlStatDisk_GetUsageDiskSpace(hDiskStatistics, &nUsedDiskSpace);
            nUsedDiskSpace /= (1024 * 1024);
            nFreeDiskSpace /= (1024 * 1024);

            printf("%7d %10lld %13lld %20lld %20lld\n",
                nCpuUsage, nUsedRam, nFreeRam, nUsedDiskSpace, nFreeDiskSpace);

            PrlHandle_Free(hDiskStatistics);
            PrlHandle_Free(hCpuStatistics);
            PrlHandle_Free(hServerStatistics);
            PrlHandle_Free(hEventParameters);
        }
    }

    PrlHandle_Free(handle);

    return PRL_ERR_SUCCESS;
}

// ------------------------------
// Program entry point.
// ------------------------------

int main(int argc, char* argv[])
{
    PRL_HANDLE hServer;
    PRL_RESULT ret;

    // Use the correct dynamic library depending on the platform.
    #ifdef _WIN_
    #define SDK_LIB_NAME "prl_sdk.dll"
    #elif defined(_LIN_)
    #define SDK_LIB_NAME "libprl_sdk.so"
    #elif defined(_MAC_)
    #define SDK_LIB_NAME "libprl_sdk.dylib"
    #endif

    if (PRL_FAILED(SdkWrap_Load(SDK_LIB_NAME)) &&
        PRL_FAILED(SdkWrap_Load("./" SDK_LIB_NAME)))
    {
        fprintf(stderr, "Failed to load " SDK_LIB_NAME "\n");
        return -1;
    }

    // Initialize the Virtuozzo API.
    ret = PrlApi_Init(PARALLELS_API_VER);
    if (PRL_FAILED(ret))
    {
        fprintf(stderr, "PrlApi_Init returned with error: %s.\n",
            PRL_RESULT_TO_STRING(ret));
        PrlApi_Deinit();
        SdkWrap_Unload();
        return ret;
    }

    // Create a server handle.
    ret = PrlSrv_Create(&hServer);
    if (PRL_FAILED(ret))
    {
        fprintf(stderr, "PrlSvr_Create failed, error: %s",
            PRL_RESULT_TO_STRING(ret));
        PrlHandle_Free(hServer);
        PrlApi_Deinit();
        SdkWrap_Unload();
        return -1;
```

```
}

// Log in (PrlSrv is asynchronous).
PRL_HANDLE hJob = PrlSrv_Login(
        hServer, // PRL_HANDLE of type PHT_SERVER.
        szServer, // Server hostname or IP address.
        szUsername, // Username.
        szPassword, // Password.
        0, // Deprecated - UUID of previous session.
        0, // Optional - port number (0 for default).
        0, // Optional - timeout value (0 for default).
        PSL_LOW_SECURITY); // Security level

ret = PrlJob_Wait(hJob, 1000);
if (PRL_FAILED(ret))
{
    fprintf(stderr, "PrlJob_Wait for PrlSrv_Login returned with error: %s\n",
        PRL_RESULT_TO_STRING(ret));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}

// Analyze the result of PrlSrv_Login.
PRL_RESULT nJobResult;
ret = PrlJob_GetRetCode(hJob, &nJobResult);
if (PRL_FAILED(nJobResult))
{
    printf("Login job returned with error: %s\n",
        PRL_RESULT_TO_STRING(nJobResult));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
    return -1;
}
else
    printf("Login was successful.\n");
// ------------------------------------------------------------------------
// 1. Register our event handler (OurCallback function).
// 2. Subscribe to host machine statistics events.
// 3. Keep receiving events until user presses <enter> key.
// 4. Unsubscribe from statistics events.
// 5. Un-register out event handler.
// ------------------------------------------------------------------------
PrlSrv_RegEventHandler(hServer, OurCallback, NULL);
PrlSrv_SubscribeToHostStatistics(hServer);
char c;
scanf(&c, 1);
PrlSrv_UnsubscribeFromHostStatistics(hServer);
PrlSrv_UnregEventHandler(hServer, OurCallback, NULL);
// ------------------------------------------------------------------------

// Log off.
hJob = PrlSrv_Logoff(hServer);
ret = PrlJob_Wait(hJob, 1000);
if (PRL_FAILED(ret))
{
    fprintf(stderr, "PrlJob_Wait for PrlSrv_Logoff returned error: %s\n",
        PRL_RESULT_TO_STRING(ret));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);
    PrlApi_Deinit();
    SdkWrap_Unload();
```

```
        return -1;
    }

    ret = PrlJob_GetRetCode(hJob, &nJobResult);
    if (PRL_FAILED(ret))
    {
        fprintf(stderr, "PrlJob_GetRetCode failed for PrlSrv_Logoff with error: %s\n",
            PRL_RESULT_TO_STRING(ret));
        PrlHandle_Free(hJob);
        PrlHandle_Free(hServer);
        PrlApi_Deinit();
        SdkWrap_Unload();
        return -1;
    }

    // Report success or failure of PrlSrv_Logoff.
    if (PRL_FAILED(nJobResult))
    {
        fprintf(stderr, "PrlSrv_Logoff failed with error: %s\n",
            PRL_RESULT_TO_STRING(nJobResult));
        PrlHandle_Free(hJob);
        PrlHandle_Free(hServer);
        PrlApi_Deinit();
        SdkWrap_Unload();
        return -1;
    }
    else
        printf("Logoff was successful.\n");

    // Free handles that are no longer required.
    PrlHandle_Free(hJob);
    PrlHandle_Free(hServer);

    // De-initialize the Virtuozzo API, and unload the SDK.
    PrlApi_Deinit();
    SdkWrap_Unload();

    return 0;
}
```

PrlSrv_SubscribeToPerfStats Function

Allows to subscribe to periodically receive perfomance statistics.

Syntax

```
PRL_HANDLE PrlSrv_SubscribeToPerfStats(
    PRL_HANDLE hServer,
    PRL_CONST_STR sFilter
);
```

File

PrlApiStat.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sFilter

String representing statistics filter, use char '*' at the end to get subset of statistic. Could be NULL, to get all statistic. See PrlPerfCounters.h header file in order to determine published performance counters patterns list.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To obtain statistics, an event handler (callback function) is required. Within the event handler, first check the type of event. Events of type PET_DSP_EVT_PERFSTATS_UPDATED indicate an event containing statistics data. The perfomace statistics repesented by parameters of Event object (a handle of type PHT_EVENT (p. 72)).

For the event handler to be called, it is necessary to register it with PrlSrv_RegEventHandler (p. 284). When statistics data is no longer required, unsubscribe from statistics events using PrlSrv_UnsubscribeFromHostPerfStats. When events are no longer required, unregister the event handler using PrlSrv_UnregEventHandler (p. 306).

If method is called more than once, the 'filter' value is updated each time on the Virtuozzo Service side.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_UnregEventHandler Function

Unregisters the specified event handler (callback function) that was previously registered with the PrlSrv_RegEventHandler (p. 284) function.

Syntax

```
PRL_RESULT PrlSrv_UnregEventHandler(
    PRL_HANDLE hServer,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR userData
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

handler

A pointer to the callback function (event handler) to unregister. This must be the same pointer that was passed to the PrlSrv_RegEventHandler (p. 284) when the event handler was registered with the virtual machine.

userData

A pointer to the user data. This must also be the same pointer that was passed to the PrlSrv_RegEventHandler (p. 284) when the event handler was registered with the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When using callback functionality with asynchronous functions, a callback function is first registered with Virtuozzo Service. When no longer needed (client doesn't want to receive events anymore), the function must by unregistered using the PrlSrv_UnregEventHandler function. The values of the handler and userData parameters constitute a unique pair which will be used to identify the event handler that will be unregistered. Make sure that you supply the same values as the ones you used when you registered the handler. See PrlSrv_SubscribeToHostStatistics (p. 301) for the complete example illustrating event handlers usage.

PrlSrv_UnsubscribeFromHostStatistics Function

Cancels the Virtuozzo Service statistics subscription that was started using the PrlSrv_SubscribeToHostStatistics (p. 301) function.

Syntax

```
PRL_HANDLE PrlSrv_UnsubscribeFromHostStatistics(
    PRL_HANDLE hServer
);
```

File

PrlApiStat.h

Parameters

handle

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_UnsubscribeFromPerfStats Function

Cancels the Virtuozzo Service perfomace statistics subscription that was started using the PrlSrv_SubscribeToPerfStats (p. 305) function.

Syntax

```
PRL_HANDLE PrlSrv_UnsubscribeFromPerfStats(
    PRL_HANDLE hServer
);
```

File

PrlApiStat.h

Parameters

handle

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_UpdateIPPrivateNetwork Function

Updates parameters of an existing private network.

Syntax

```
PRL_HANDLE PrlSrv_UpdateIPPrivateNetwork(
    PRL_HANDLE hServer,
    PRL_HANDLE hPrivNet,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hPrivNet

A handle of type PHT_IPPRIV_NET (p. 133) identifying a private network.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_AddIPPrivateNetwork (p. 223)

PrlSrv_GetIPPrivateNetworksList (p. 248)

PrlSrv_DeleteIPPrivateNetwork

PrlSrv_UpdateLicense Function

Installs Virtuozzo license on the specified Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_UpdateLicense(
    PRL_HANDLE hServer,
    PRL_CONST_STR sKey,
    PRL_CONST_STR sUser,
    PRL_CONST_STR sCompany
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

sKey

License key.

sUser

License user name.

sCompany

License company name.

nFlags

Flags (from enums PRL_UPDATE_LICENSE_FLAGS (p. 1097) + PRL_API_COMMAND_FLAGS (p. 1037))

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to

the PHT_RESULT (p. 210) object.

2. Use the PrlResult_GetParam (p. 210) function to obtain a handle

of type PHT_LICENSE (p. 146) containing the license information.

PrlSrv_UpdateLicenseEx Function

Syntax

```
PRL_HANDLE PrlSrv_UpdateLicenseEx(
    PRL_HANDLE hServer,
    PRL_CONST_STR sKey,
    PRL_CONST_STR sUser,
    PRL_CONST_STR sCompany,
    PRL_UINT32 nFlags
);
```

File

PrlApiDisp.h

PrlSrv_UpdateNetworkClassesList Function

Update network class classes configuration.

Syntax

```
PRL_HANDLE PrlSrv_UpdateNetworkClassesList(
    PRL_HANDLE hServer,
    PRL_HANDLE hNetworkClassesList,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

nFlags

Reserved parameter.

hNetworkClassList

A handle of type PHT_HANDLES_LIST (p. 108) contains the list of handles of type PHT_NETWORK_CLASS (p. 154)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_GetNetworkClassesList (p. 250)

PrlSrv_UpdateNetworkShapingConfig Function

Update the network shaping configuration.

Syntax

```
PRL_HANDLE PrlSrv_UpdateNetworkShapingConfig(
    PRL_HANDLE hServer,
    PRL_HANDLE hNetworkShapingConfig,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hNetworkShapingConfig

A handle of type PHT_NETWORK_SHAPING_CONFIG (p. 169).

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_GetNetworkShapingConfig (p. 251)

PrlSrv_UpdateOfflineService Function

Add/Updates parameters of an offline management service.

Syntax

```
PRL_HANDLE PrlSrv_UpdateOfflineService(
    PRL_HANDLE hServer,
    PRL_HANDLE hOffmgmtService,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hOffmgmtService

A handle of type PHT_OFFLINE_SERVICE (p. 173) identifying a offline management service

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_DeleteOfflineService (p. 234)

PrlSrv_GetOfflineServicesList (p. 252)

PrlSrv_UpdateVirtualNetwork Function

Updates parameters of an existing virtual network.

Syntax

```
PRL_HANDLE PrlSrv_UpdateVirtualNetwork(
    PRL_HANDLE hServer,
    PRL_HANDLE hVirtNet,
    PRL_UINT32 nFlags
);
```

File

PrlApiNet.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying a virtual network.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_AddVirtualNetwork (p. 224)

PrlSrv_GetVirtualNetworkList (p. 263)

PrlSrv_DeleteVirtualNetwork (p. 235)

PrlSrv_UserProfileBeginEdit Function

Timestamps the beginning of the user profile editing operation.

Syntax

```
PRL_HANDLE PrlSrv_UserProfileBeginEdit(
    PRL_HANDLE hServer
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Every modification to the user profile (see PHT_USER_PROFILE (p. 374)) must begin with this function call. The function, together with the PrlSrv_UserProfileCommit (p. 314) function, is used to detect collisions with other clients trying to modify the profile of the same user at the same time. The function does not lock the profile being edited. If a collision is detected on commit, the changes are rejected.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_ACCESS_DENIED (p. 1175) - the user does not have sufficient rights to perform this task.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrv_UserProfileCommit Function

Saves (commits) user profile changes to the Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlSrv_UserProfileCommit(
    PRL_HANDLE hServer,
    PRL_HANDLE hUserProfile
);
```

File

PrlApiDisp.h

Parameters

hServer

A handle of type PHT_SERVER (p. 219) identifying the Virtuozzo Service.

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) containing the user information.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Every modification to the user profile (see PHT_USER_PROFILE (p. 374)) must begin with the PrlSrv_UserProfileBeginEdit (p. 313) function call. The function, together with the PrlSrv_UserProfileCommit function, is used to detect collisions with other client trying to modify the profile of the same user at the same time. The function does not lock the profile being edited. If a collision is detected on commit, the changes are rejected. Otherwise, the changes are saved to the Virtuozzo Service.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_SERVER_CONFIG

The PHT_SERVER_CONFIG handle is used to obtain the host computer configuration information. The handle is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetSrvConfig (p. 257) function call. The handle can also be used to obtain network settings of a guest operating system installed in a virtual machine. In this case, the handle is obtained using the PrlVmGuest_GetNetworkSettings (p. 716) function.

Functions

	Name	Description
	PrlSrvCfg_GetCpuCount (p. 316)	Determines the number of CPUs on the host machine.
	PrlSrvCfg_GetCpuFeatures (p. 317)	To get the CPU features from server hSrvConfig : A handle of type PRL_HANDLE (p. 820).
	PrlSrvCfg_GetCpuFeaturesEx (p. 317)	Allows to get the CPU features from server.
	PrlSrvCfg_GetCpuFeaturesMaskingCapabilities (p. 318)	Allows to get the CPU features masking capabilities from server
	PrlSrvCfg_GetCpuHvt (p. 318)	Determines the hardware virtualization type of the host CPU.
	PrlSrvCfg_GetCpuMode (p. 319)	Determines the CPU mode (32 bit or 64 bit) on the host.

	PrlSrvCfg_GetCpuModel (p. 319)	Determines the model of CPU on the host machine.
	PrlSrvCfg_GetCpuSpeed (p. 320)	Determines the host machine CPU speed.
	PrlSrvCfg_GetDefaultGateway (p. 320)	Obtains the global default gateway address of the specified host or guest.
	PrlSrvCfg_GetDefaultGatewayIPv6 (p. 321)	Obtains the global default IPv6 gateway address of the specified host or guest.
	PrlSrvCfg_GetDnsServers (p. 322)	Obtains the list of IP addresses of DNS servers for the specified host or guest.
	PrlSrvCfg_GetFloppyDisk (p. 322)	Obtains an object of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a floppy disk drive on the host.
	PrlSrvCfg_GetFloppyDisksCount (p. 323)	Determines the number of floppy disk drives on the host.
	PrlSrvCfg_GetGenericPciDevice (p. 323)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a PCI device installed on the host.
	PrlSrvCfg_GetGenericPciDevicesCount (p. 324)	Determines the number of PCI devices installed on the host.
	PrlSrvCfg_GetGenericScsiDevice (p. 325)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a generic SCSI device installed on the host.
	PrlSrvCfg_GetGenericScsiDevicesCount (p. 325)	Determines the number of generic SCSI devices installed on the host.
	PrlSrvCfg_GetHardDisk (p. 326)	Obtains an object of type PHT_HW_HARD_DISK (p. 117) containing information about a hard disks drive on the host.
	PrlSrvCfg_GetHardDisksCount (p. 327)	Determines the number of hard disk drives on the host.
	PrlSrvCfg_GetHostname (p. 327)	Returns the hostname of the specified host or guest.
	PrlSrvCfg_GetHostOsMajor (p. 328)	Returns the major version number of the host operating system.
	PrlSrvCfg_GetHostOsMinor (p. 328)	Returns the minor version number of the host operating system.
	PrlSrvCfg_GetHostOsStrPresentation (p. 329)	Returns the full host operating system information as a single string.
	PrlSrvCfg_GetHostOsSubMinor (p. 329)	Returns the sub-minor version number of the host operating system.
	PrlSrvCfg_GetHostOsType (p. 330)	Returns the host operating system type.
	PrlSrvCfg_GetHostRamSize (p. 331)	Determines the amount of memory (RAM) available on the host.
	PrlSrvCfg_GetMaxHostNetAdapters (p. 331)	Returns max possible Virtuozzo network adapters on the host.
	PrlSrvCfg_GetMaxVmNetAdapters (p. 332)	Returns max possible Virtuozzo virtual network adapters which can be applied per VM.
	PrlSrvCfg_GetNetAdapter (p. 332)	Obtains a handle of type PHT_HW_NET_ADAPTER (p. 126) containing information about a network adapter installed in the host or guest.
	PrlSrvCfg_GetNetAdaptersCount (p. 333)	Determines the number of network adapters available on the server.
	PrlSrvCfg_GetOpticalDisk (p. 333)	Obtains an object of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about an optical disk on the host.
	PrlSrvCfg_GetOpticalDisksCount (p. 334)	Determines the number of optical disks on the host.
	PrlSrvCfg_GetParallelPort (p. 334)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a printer port on the host.
	PrlSrvCfg_GetParallelPortsCount (p. 335)	Determines the number of printer ports on the host.
	PrlSrvCfg_GetPrinter (p. 336)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a printer installed on the host.
	PrlSrvCfg_GetPrintersCount (p. 336)	Determines the number of printers installed on the host.
	PrlSrvCfg_GetSearchDomains (p. 337)	Obtains the list of search domains for the specified host or guest.

	PrlSrvCfg_GetSerialPort (p. 337)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a serial port on the host.
	PrlSrvCfg_GetSerialPortsCount (p. 338)	Determines the number of serial ports available on the host.
	PrlSrvCfg_GetSoundMixerDev (p. 338)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a sound mixer device on the host.
	PrlSrvCfg_GetSoundMixerDevsCount (p. 339)	Determines the number of sound mixer devices available on the host.
	PrlSrvCfg_GetSoundOutputDev (p. 340)	Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a sound device on the host.
	PrlSrvCfg_GetSoundOutputDevsCount (p. 340)	Determines the number of sound devices available on the host.
	PrlSrvCfg_GetUsbDev (p. 341)	Obtains a pointer of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a USB device on the host.
	PrlSrvCfg_GetUsbDevsCount (p. 341)	Determines the number of USB devices on the host.
	PrlSrvCfg_IsSoundDefaultEnabled (p. 342)	Determines whether a sound device on the host is enabled or disabled.
	PrlSrvCfg_IsUsbSupported (p. 342)	Determines if USB is supported on the host.
	PrlSrvCfg_IsVtdSupported (p. 343)	Determines whether VT-d is supported on the host.
	PrlSrvCfgPci_IsPrimaryDevice (p. 343)	Determines whether a specified PCI device (network adapter, display adapter, sound card, etc.) is the primary adapter in the system.

PrlSrvCfg_GetCpuCount Function

Determines the number of CPUs on the host machine.

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnCpuCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnCpuCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetCpuFeatures Function

To get the CPU features from server hSrvConfig : A handle of type PRL_HANDLE (p. 820).

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuFeatures(
    PRL_HANDLE hSrvConfig,
    PRL_CPU_FEATURES_PTR pCpuFeatures
);
```

File

PrlApiDisp.h

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

pMasks: [out] A pointer structure with processor features set

PrlSrvCfg_GetCpuFeaturesEx Function

Allows to get the CPU features from server.

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuFeaturesEx(
    PRL_HANDLE hSrvConfig,
    PRL_HANDLE_PTR phCpuFeatures
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

phCpuFeatures

A pointer to a variable that receives a handle of type PHT_CPU_FEATURES (p. 22).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate new object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetCpuFeaturesMaskingCapabilities Function

Allows to get the CPU features masking capabilities from server

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuFeaturesMaskingCapabilities(
    PRL_HANDLE hSrvConfig,
    PRL_HANDLE_PTR phCpuFeatures
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

phCpuFeatures

A pointer to a variable that receives a handle of type PHT_CPU_FEATURES (p. 22).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate new object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetCpuHvt Function

Determines the hardware virtualization type of the host CPU.

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuHvt(
    PRL_HANDLE hSrvConfig,
    PRL_CPU_HVT_PTR pnCpuHvt
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnCpuMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetCpuMode Function

Determines the CPU mode (32 bit or 64 bit) on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuMode(
    PRL_HANDLE hSrvConfig,
    PRL_CPU_MODE_PTR pnCpuMode
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnCpuMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetCpuModel Function

Determines the model of CPU on the host machine.

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuModel(
    PRL_HANDLE hSrvConfig,
    PRL_STR sCpuModel,
    PRL_UINT32_PTR pnCpuModelBufLength
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

sCpuModel

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnCpuModelBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetCpuSpeed Function

Determines the host machine CPU speed.

Syntax

```
PRL_RESULT PrlSrvCfg_GetCpuSpeed(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnCpuSpeed
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnCpuSpeed

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetDefaultGateway Function

Obtains the global default gateway address of the specified host or guest.

Syntax

```
PRL_RESULT PrlSrvCfg_GetDefaultGateway(
    PRL_HANDLE hSrvConfig,
    PRL_STR sDefaultGateway,
    PRL_UINT32_PTR pnDefaultGatewayBufLength
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314). When obtaining the host OS gateway address, use the PrlSrv_GetSrvConfig (p. 257) to obtain the handle. When obtaining the guest OS gateway address, use the PrlVmGuest_GetNetworkSettings (p. 716) function.

sDefaultGateway

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetDefaultGatewayIPv6 Function

Obtains the global default IPv6 gateway address of the specified host or guest.

Syntax

```
PRL_RESULT PrlSrvCfg_GetDefaultGatewayIPv6(
    PRL_HANDLE hSrvConfig,
    PRL_STR sDefaultGateway,
    PRL_UINT32_PTR pnDefaultGatewayBufLength
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314). When obtaining the host OS gateway address, use the PrlSrv_GetSrvConfig (p. 257) to obtain the handle. When obtaining the guest OS gateway address, use the PrlVmGuest_GetNetworkSettings (p. 716) function.

sDefaultGateway

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetDnsServers Function

Obtains the list of IP addresses of DNS servers for the specified host or guest.

Syntax

```
PRL_RESULT PrlSrvCfg_GetDnsServers(
    PRL_HANDLE hSrvConfig,
    PRL_HANDLE_PTR phDnsServersList
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314). When obtaining the host OS information, use the PrlSrv_GetSrvConfig (p. 257) to obtain the handle. When obtaining the guest OS information, use the PrlVmGuest_GetNetworkSettings (p. 716) function.

phDnsServersList

[out] A pointer to a variable that receives the PHT_STRINGS_LIST (p. 357) handle containing the IP addresses of DNS servers.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate strings list object

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetFloppyDisk Function

Obtains an object of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a floppy disk drive on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetFloppyDisk(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the drive (beginning with 0). The number of drives in the list can be determined by making the PrlSrvCfg_GetFloppyDisksCount (p. 323) function call.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the drive information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetFloppyDisksCount Function

Determines the number of floppy disk drives on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetFloppyDisksCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnFloppyDisksCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnFloppyDisksCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetGenericPciDevice Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a PCI device installed on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetGenericPciDevice(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the GenericPciDevice (beginning with 0). To get the number of installed generic PCI devices, use the PrlSrvCfg_GetGenericPciDevicesCount (p. 324) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the generic PCI device information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrvCfg_GetGenericPciDevicesCount (p. 324)

PrlSrvCfg_GetGenericPciDevicesCount Function

Determines the number of PCI devices installed on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetGenericPciDevicesCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnGenericPciDevicesCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnGenericPciDevicesCount

[out] A pointer to a variable that receives the number of devices

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrvCfg_GetGenericPciDevice (p. 323)

PrlSrvCfg_GetGenericScsiDevice Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a generic SCSI device installed on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetGenericScsiDevice(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the GenericScsiDevice (beginning with 0). To get the number of installed generic SCSI devices, use the PrlSrvCfg_GetGenericScsiDevicesCount (p. 325) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the generic SCSI device information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetGenericScsiDevicesCount Function

Determines the number of generic SCSI devices installed on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetGenericScsiDevicesCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnGenericScsiDevicesCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnGenericScsiDevicesCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHardDisk Function

Obtains an object of type PHT_HW_HARD_DISK (p. 117) containing information about a hard disks drive on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHardDisk(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the drive (beginning with 0). The number of drives in the list can be determined by making the PrlSrvCfg_GetHardDisksCount (p. 327) function call.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_HARD_DISK (p. 117) containing the drive information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHardDisksCount Function

Determines the number of hard disk drives on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHardDisksCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnHardDisksCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnHardDisksCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostname Function

Returns the hostname of the specified host or guest.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostname(
    PRL_HANDLE hSrvConfig,
    PRL_STR sHostname,
    PRL_UINT32_PTR pnHostnameBufLength
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314). When obtaining the host OS information, use the PrlSrv_GetSrvConfig (p. 257) to obtain the handle. When obtaining the guest OS information, use the PrlVmGuest_GetNetworkSettings (p. 716) function.

sHostname

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnHostnameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostOsMajor Function

Returns the major version number of the host operating system.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostOsMajor(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnHostOsMajor
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnHostOsMajor

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostOsMinor Function

Returns the minor version number of the host operating system.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostOsMinor(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnHostOsMinor
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnHostOsMinor

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostOsStrPresentation Function

Returns the full host operating system information as a single string.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostOsStrPresentation(
    PRL_HANDLE hSrvConfig,
    PRL_STR sHostOsStrPresentation,
    PRL_UINT32_PTR pnHostOsStrPresentationBufLength
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

sHostOsStrPresentation

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnHostOsStrPresentationBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostOsSubMinor Function

Returns the sub-minor version number of the host operating system.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostOsSubMinor(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnHostOsSubMinor
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnHostOsSubMinor

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostOsType Function

Returns the host operating system type.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostOsType(
    PRL_HANDLE hSrvConfig,
    PRL_HOST_OS_TYPE_PTR pnHostOsType
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnHostOsType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetHostRamSize Function

Determines the amount of memory (RAM) available on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetHostRamSize(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnHostRamSize
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnHostRamSize

[out] A pointer to a variable that receives the result. The memory size is specified in Megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetMaxHostNetAdapters Function

Returns max possible Virtuozzo network adapters on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetMaxHostNetAdapters(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnMaxNetAdaptersCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnMaxNetAdaptersCount

[out] A pointer to a variable that receives the result. Null value (0) means that remote Virtuozzo server doesn't supported this interface.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetMaxVmNetAdapters Function

Returns max possible Virtuozzo virtual network adapters which can be applied per VM.

Syntax

```
PRL_RESULT PrlSrvCfg_GetMaxVmNetAdapters(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnMaxNetAdaptersCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnMaxNetAdaptersCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetNetAdapter Function

Obtains a handle of type PHT_HW_NET_ADAPTER (p. 126) containing information about a network adapter installed in the host or guest.

Syntax

```
PRL_RESULT PrlSrvCfg_GetNetAdapter(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314). When obtaining the host OS information, use the PrlSrv_GetSrvConfig (p. 257) to obtain the handle. When obtaining the guest OS information, use the PrlVmGuest_GetNetworkSettings (p. 716) function.

nIndex

A numeric index identifying the network adapter (beginning with 0). To get the number of adapters available, use the PrlSrvCfg_GetNetAdaptersCount (p. 333) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_NET_ADAPTER (p. 126) containing the network adapter information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetNetAdaptersCount Function

Determines the number of network adapters available on the server.

Syntax

```
PRL_RESULT PrlSrvCfg_GetNetAdaptersCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnNetAdaptersCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnNetAdaptersCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetOpticalDisk Function

Obtains an object of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about an optical disk on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetOpticalDisk(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the optical disk (beginning with 0). To get the number of disks available, use the PrlSrvCfg_GetOpticalDisksCount (p. 334) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the disk information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetOpticalDisksCount Function

Determines the number of optical disks on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetOpticalDisksCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnOpticalDisksCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnOpticalDisksCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetParallelPort Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a printer port on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetParallelPort(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the port (beginning with 0). To get the number ports available, use the PrlSrvCfg_GetParallelPortsCount (p. 335) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the port information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetParallelPortsCount Function

Determines the number of printer ports on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetParallelPortsCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnParallelPortsCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnParallelPortsCount

[out] A pointer to a variable that receives the result

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetPrinter Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a printer installed on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetPrinter(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the printer (beginning with 0). To get the number of installed printers, use the PrlSrvCfg_GetPrintersCount (p. 336) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the printer information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetPrintersCount Function

Determines the number of printers installed on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetPrintersCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnPrintersCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnPrintersCount

[out] A pointer to a variable that receives the

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSearchDomains Function

Obtains the list of search domains for the specified host or guest.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSearchDomains(
    PRL_HANDLE hSrvConfig,
    PRL_HANDLE_PTR phSearchDomainsList
);
```

File

PrlApiDisp.h

Parameters

phSearchDomainsList

[out] A pointer to a variable that receives the PHT_STRINGS_LIST (p. 357) handle containing the list of search domains.

hNetAdapter

A handle of type PHT_SERVER_CONFIG (p. 314). When obtaining the host OS information, use the PrlSrv_GetSrvConfig (p. 257) to obtain the handle. When obtaining the guest OS information, use the PrlVmGuest_GetNetworkSettings (p. 716) function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate strings list object

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSerialPort Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a serial port on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSerialPort(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the port (beginning with 0). To get the number of ports available, use the PrlSrvCfg_GetSerialPortsCount (p. 338) function.

phDevice

[out] A pointer to a variable that receives a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing the port information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSerialPortsCount Function

Determines the number of serial ports available on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSerialPortsCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnSerialPortsCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnSerialPortsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSoundMixerDev Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a sound mixer device on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSoundMixerDev(
```

```
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the sound mixer device (beginning with 0). To get the number of devices available, use the PrlSrvCfg_GetSoundMixerDevsCount (p. 339) function.

phDevice

[out] A pointer to a variable that receives a pointer of type PHT_HW_GENERIC_DEVICE (p. 111) containing the device information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSoundMixerDevsCount Function

Determines the number of sound mixer devices available on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSoundMixerDevsCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnSoundMixerDevsCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnSoundMixerDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSoundOutputDev Function

Obtains a handle of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a sound device on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSoundOutputDev(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the sound device (beginning with 0). To get the number of sound devices available, use the PrlSrvCfg_GetSoundOutputDevsCount (p. 340) function.

phDevice

[out] A pointer to a variable that receives a pointer of type PHT_HW_GENERIC_DEVICE (p. 111) containing the device information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetSoundOutputDevsCount Function

Determines the number of sound devices available on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetSoundOutputDevsCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnSoundOutputDevsCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnSoundOutputDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetUsbDev Function

Obtains a pointer of type PHT_HW_GENERIC_DEVICE (p. 111) containing information about a USB device on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetUsbDev(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nIndex

A numeric index identifying the device (beginning with 0). To get the number of USB devices available, use the PrlSrvCfg_GetUsbDevsCount (p. 341) function.

phDevice

[out] A pointer to a variable that receives a pointer of type PHT_HW_GENERIC_DEVICE (p. 111) containing the device information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_GetUsbDevsCount Function

Determines the number of USB devices on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_GetUsbDevsCount(
    PRL_HANDLE hSrvConfig,
    PRL_UINT32_PTR pnUsbDevsCount
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pnUsbDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_IsSoundDefaultEnabled Function

Determines whether a sound device on the host is enabled or disabled.

Syntax

```
PRL_RESULT PrlSrvCfg_IsSoundDefaultEnabled(
    PRL_HANDLE hSrvConfig,
    PRL_BOOL_PTR pbSoundDefaultEnabled
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pbSoundDefaultEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the device is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_IsUsbSupported Function

Determines if USB is supported on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_IsUsbSupported(
    PRL_HANDLE hSrvConfig,
    PRL_BOOL_PTR pbUsbSupported
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pbUsbSupported

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that USB is supported. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfg_IsVtdSupported Function

Determines whether VT-d is supported on the host.

Syntax

```
PRL_RESULT PrlSrvCfg_IsVtdSupported(
    PRL_HANDLE hSrvConfig,
    PRL_BOOL_PTR pbVtdSupported
);
```

File

PrlApiDisp.h

Parameters

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

pbVtdSupported

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that VT-d is supported. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvCfgPci_IsPrimaryDevice Function

Determines whether a specified PCI device (network adapter, display adapter, sound card, etc.) is the primary adapter in the system.

Syntax

```
PRL_RESULT PrlSrvCfgPci_IsPrimaryDevice(
    PRL_HANDLE hGenericPci,
    PRL_BOOL_PTR pbIsPrimaryDevice
);
```

File

PrlApiDisp.h

Parameters

hGenericPci

A valid handle of type PHT_HW_GENERIC_PCI_DEVICE (p. 115) identifying the adapter.

pbIsPrimaryDevice

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the PCI device is primary.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_SERVER_INFO

The PHT_SERVER_INFO handle is a container that contains information about a host machine. The information includes the Virtuozzo Service port number, host name, OS version, product version, and host machine UUID. To obtain this information for the specified Virtuozzo Service, use the PrlSrv_GetServerInfo (p. 256) function. You can also search for Virtuozzo Services running on your network by executing the PrlSrv_LookupParallelsServers (p. 279) function. The function returns a list of PHT_SERVER_INFO handles for the Virtuozzo Services that it finds. The returned information can be used to establish a connection with the Virtuozzo Service of interest.

Functions

	Name	Description
	PrlSrvInfo_GetApplicationMode (p. 345)	Returns the application mode of server.
	PrlSrvInfo_GetCmdPort (p. 345)	Returns the port number at which the Virtuozzo Service is listening for requests.
	PrlSrvInfo_GetHostName (p. 346)	Returns the name of the machine hosting the specified Virtuozzo Service.
	PrlSrvInfo_GetOsVersion (p. 346)	Returns the version of the host operating system.
	PrlSrvInfo_GetProductVersion (p. 347)	Returns the Virtuozzo product version number.
	PrlSrvInfo_GetServerUuid (p. 348)	Returns the host machine UUID (universally unique ID).
	PrlSrvInfo_GetStartTime (p. 348)	Returns the service start time in milliseconds since Epoch (01-01-1970).
	PrlSrvInfo_GetStartTimeMonotonic (p. 349)	Returns the service start time in milliseconds (monotonic time clock).

PrlSrvInfo_GetApplicationMode Function

Returns the application mode of server.

Syntax

```
PRL_RESULT PrlSrvInfo_GetApplicationMode(
    PRL_HANDLE hServerInfo,
    PRL_APPLICATION_MODE_PTR pnApplicationMode
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

pnAppMode

[out] A pointer to a variable that receives the server application mode.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvInfo_GetCmdPort Function

Returns the port number at which the Virtuozzo Service is listening for requests.

Syntax

```
PRL_RESULT PrlSrvInfo_GetCmdPort(
    PRL_HANDLE hServerInfo,
    PRL_UINT32_PTR pnPort
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

pnPort

[out] A pointer to a variable that receives the port number.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvInfo_GetHostName Function

Returns the name of the machine hosting the specified Virtuozzo Service.

Syntax

```
PRL_RESULT PrlSrvInfo_GetHostName(
    PRL_HANDLE hServerInfo,
    PRL_STR sHostName,
    PRL_UINT32_PTR pnHostNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

sHostName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnHostNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvInfo_GetOsVersion Function

Returns the version of the host operating system.

Syntax

```
PRL_RESULT PrlSrvInfo_GetOsVersion(
    PRL_HANDLE hServerInfo,
    PRL_STR sOsVersion,
    PRL_UINT32_PTR pnOsVersionBufLength
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

sOsVersion

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnOsVersionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvInfo_GetProductVersion Function

Returns the Virtuozzo product version number.

Syntax

```
PRL_RESULT PrlSrvInfo_GetProductVersion(
    PRL_HANDLE hServerInfo,
    PRL_STR sProductVersion,
    PRL_UINT32_PTR pnProductVersionBufLength
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

sProductVersion

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnProductVersionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvInfo_GetServerUuid Function

Returns the host machine UUID (universally unique ID).

Syntax

```
PRL_RESULT PrlSrvInfo_GetServerUuid(
    PRL_HANDLE hServerInfo,
    PRL_STR sServerUuid,
    PRL_UINT32_PTR pnServerUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

sServerUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnServerUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlSrvInfo_GetStartTime Function

Returns the service start time in milliseconds since Epoch (01-01-1970).

Syntax

```
PRL_RESULT PrlSrvInfo_GetStartTime(
    PRL_HANDLE hServerInfo,
```

```
    PRL_UINT64_PTR pnStartTime
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

pnStartTime

[out] A pointer to a variable that receives the service start time in milliseconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrvInfo_GetStartTimeMonotonic (p. 349) PrlHandle_GetPackageId (p. 750)

PrlSrvInfo_GetStartTimeMonotonic Function

Returns the service start time in milliseconds (monotonic time clock).

Syntax

```
PRL_RESULT PrlSrvInfo_GetStartTimeMonotonic(
    PRL_HANDLE hServerInfo,
    PRL_UINT64_PTR pnStartTimeMonotonic
);
```

File

PrlApiDisp.h

Parameters

hServerInfo

A handle of type PHT_SERVER_INFO (p. 344) containing the Virtuozzo Service information.

pnStartTimeMonotonic

[out] A pointer to a variable that receives the service start time in milliseconds from monotonic time clock.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the PHT_SERVER_INFO (p. 344) object does not contain any data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrvInfo_GetStartTime (p. 348) PrlHandle_GetPackageId (p. 750)

PHT_SESSION_INFO

PHT_SHARE

The PHT_SHARE handle is used to manage host shares. With this handle, you can make a host OS folder visible and accessible in a virtual machine. The handle is obtained by making the PrlVmCfg_GetShare (p. 603) or the PrlVmCfg_CreateShare (p. 550) function call.

Description

To create a new share:

1. Call the PrlVm_BeginEdit (p. 446) function to mark the beginning of the operation (this is essentially a virtual machine configuration modification, so it must be done within the begin/commit block).
2. Use the PrlVmCfg_CreateShare (p. 550) function to obtain the handle.
3. Use functions provided by this object to set the desired new share properties.
4. Call the PrlVm_Commit (p. 453) to finalize the changes.
5. Enabled the host sharing feature in the virtual machine by calling the PrlVm_SetHostSharingEnabled function.

To modify an existing share, use the PrlVmCfg_GetShare (p. 603) function to obtain a handle containing the share information. Then repeat the steps described above.

Functions

	Name	Description
	PrlShare_GetDescription (p. 351)	Returns the shared folder description.
	PrlShare_GetName (p. 351)	Returns the shared folder name (as it appears in the guest OS).
	PrlShare_GetPath (p. 352)	Returns the shared folder path.
	PrlShare_IsEnabled (p. 353)	Determines whether the share is enabled or not.
	PrlShare_IsReadOnly (p. 353)	Determines if the share is read-only.
	PrlShare_Remove (p. 354)	Removes the share from the virtual machine configuration.
	PrlShare_SetDescription (p. 354)	Sets the shared folder description.
	PrlShare_SetEnabled (p. 355)	Enables the specified share.
	PrlShare_SetName (p. 356)	Sets the share name (as it will appear in the guest OS).
	PrlShare_SetPath (p. 356)	Sets the shared folder path.
	PrlShare_SetReadOnly (p. 357)	Makes the specified shared folder read-only.

PrlShare_GetDescription Function

Returns the shared folder description.

Syntax

```
PRL_RESULT PrlShare_GetDescription(
    PRL_HANDLE hVmShare,
    PRL_STR sShareDescription,
    PRL_UINT32_PTR pnShareDescriptionBufLength
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

sShareDescription

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnShareDescriptionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_SetDescription (p. 354)

PrlShare_GetName Function

Returns the shared folder name (as it appears in the guest OS).

Syntax

```
PRL_RESULT PrlShare_GetName(
    PRL_HANDLE hVmShare,
    PRL_STR sShareName,
    PRL_UINT32_PTR pnShareNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

sShareName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnShareNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_SetName (p. 356)

PrlShare_GetPath Function

Returns the shared folder path.

Syntax

```
PRL_RESULT PrlShare_GetPath(
    PRL_HANDLE hVmShare,
    PRL_STR sSharePath,
    PRL_UINT32_PTR pnSharePathBufLength
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

sSharePath

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnSharePathBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_SetPath (p. 356)

PrlShare_IsEnabled Function

Determines whether the share is enabled or not.

Syntax

```
PRL_RESULT PrlShare_IsEnabled(
    PRL_HANDLE hVmShare,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the share is enabled. PRL_FALSE (p. 1626) indicates that it is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_SetEnabled (p. 355)

PrlShare_IsReadOnly Function

Determines if the share is read-only.

Syntax

```
PRL_RESULT PrlShare_IsReadOnly(
    PRL_HANDLE hVmShare,
    PRL_BOOL_PTR pbReadOnly
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

pbReadOnly

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the share is read-only (files can be read and copied but not modified). PRL_FALSE (p. 1626) indicates that the folder contents can be modified.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_SetReadOnly (p. 357)

PrlShare_Remove Function

Removes the share from the virtual machine configuration.

Syntax

```
PRL_RESULT PrlShare_Remove(
    PRL_HANDLE hVmShare
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Once a share is removed from the virtual machine configuration, the folder cannot be accessed from the guest OS. Just like any other virtual machine configuration modification, this operation must be performed within the PrlVm_BeginEdit (p. 446) / PrlVm_Commit (p. 453) block.

PrlShare_SetDescription Function

Sets the shared folder description.

Syntax

```
PRL_RESULT PrlShare_SetDescription(
    PRL_HANDLE hVmShare,
    PRL_CONST_STR sNewShareDescription
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

sNewShareDescription

A user-defined share description (must be a UTF-8 encoded, null-terminated string).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_GetDescription (p. 351)

PrlShare_SetEnabled Function

Enables the specified share.

Syntax

```
PRL_RESULT PrlShare_SetEnabled(
    PRL_HANDLE hVmShare,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

bEnabled

Specify PRL_TRUE (p. 1652) to enable the share. Specified PRL_FALSE (p. 1626) to disable it. If a share is disabled, it cannot be accessed from the guest OS.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_IsEnabled (p. 353)

PrlShare_SetName Function

Sets the share name (as it will appear in the guest OS).

Syntax

```
PRL_RESULT PrlShare_SetName(
    PRL_HANDLE hVmShare,
    PRL_CONST_STR sNewShareName
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

sNewShareName

The user-defined share name (must be a a UTF-8 encoded, null-terminated string).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_GetName (p. 351)

PrlShare_SetPath Function

Sets the shared folder path.

Syntax

```
PRL_RESULT PrlShare_SetPath(
    PRL_HANDLE hVmShare,
    PRL_CONST_STR sNewSharePath
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

sNewSharePath

A string containing an absolute path to the folder on the host that you would like to share with the guest OS (must be a UTF-8 encoded, null-terminated string).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_GetPath (p. 352)

PrlShare_SetReadOnly Function

Makes the specified shared folder read-only.

Syntax

```
PRL_RESULT PrlShare_SetReadOnly(
    PRL_HANDLE hVmShare,
    PRL_BOOL bReadOnly
);
```

File

PrlApiVm.h

Parameters

hVmShare

A handle of type PHT_SHARE (p. 350) identifying the shared folder.

bReadOnly

Specify PRL_TRUE (p. 1652) to make the share read-only. Specify PRL_FALSE (p. 1626) to make the share editable.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlShare_IsReadOnly (p. 353)

PHT_STRINGS_LIST

The PHT_STRING_LIST handle is a generic string container. A new handle is created by making the PrlApi_CreateStringsList (p. 730) function call.

Functions

	Name	Description
	PrlStrList_AddItem (p. 358)	Adds a new string to the container.
	PrlStrList_GetItem (p. 358)	Extracts the specified string from the container.
	PrlStrList_GetItemsCount (p. 359)	Determines the number of strings in the container.
	PrlStrList_RemoveItem (p. 360)	Removes the specified string from the container.

PrlStrList_AddItem Function

Adds a new string to the container.

Syntax

```
PRL_RESULT PrlStrList_AddItem(
    PRL_HANDLE hStringsList,
    PRL_CONST_STR sItem
);
```

File

PrlApiCore.h

Parameters

hStringsList

A handle of type PHT_STRINGS_LIST (p. 357) identifying the string container.

sItem

A pointer to the string to add to the container. The value must be a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStrList_GetItem Function

Extracts the specified string from the container.

Syntax

```
PRL_RESULT PrlStrList_GetItem(
    PRL_HANDLE hStringsList,
    PRL_UINT32 nItemIndex,
    PRL_STR sItemBuf,
    PRL_UINT32_PTR pnItemBufSize
);
```

File

PrlApiCore.h

Parameters

hStringsList

A handle of type PHT_STRINGS_LIST (p. 357) identifying the string container.

nItemIndex

A numeric index identifying the string (beginning with 0). To determine the number of string in the container, use the PrlStrList_GetItemsCount (p. 359) function.

sItemBuf

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnItemBufSize

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStrList_GetItemsCount Function

Determines the number of strings in the container.

Syntax

```
PRL_RESULT PrlStrList_GetItemsCount(
    PRL_HANDLE hStringsList,
    PRL_UINT32_PTR pnItemsCount
);
```

File

PrlApiCore.h

Parameters

hStringsList

A handle of type PHT_STRINGS_LIST (p. 357) identifying the string container.

pnItemsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStrList_RemoveItem Function

Removes the specified string from the container.

Syntax

```
PRL_RESULT PrlStrList_RemoveItem(
    PRL_HANDLE hStringsList,
    PRL_UINT32 nItemIndex
);
```

File

PrlApiCore.h

Parameters

hStringsList

A handle of type PHT_STRINGS_LIST (p. 357) identifying the string container.

nItemIndex

A numeric index identifying the string. Index is a sequential number assigned to every item in the container. To determine the index of an item, call the PrlStrList_GetItem (p. 358) in a loop (from 0 to PrlStrList_GetItemsCount (p. 359)()). The iteration number is the index of the corresponding item.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_SYSTEM_STATISTICS

The PHT_SYSTEM_STATISTICS handle is used to obtain performance statistics for the host computer or an individual virtual machine.

Description

There are two main methods for obtaining a handle of type PHT_SYSTEM_STATISTICS:

1. Using PrlSrv_GetStatistics (p. 258) (for host statistics) or PrlVm_GetStatistics (p. 469) (for virtual machine statistics).
2. Using PrlSrv_SubscribeToHostStatistics (p. 301) (for host statistics) or PrlVm_SubscribeToGuestStatistics (p. 495) (for virtual machine statistics) to receive statistics events, and PrlSrv_UnsubscribeFromHostStatistics (p. 307) or PrlVm_UnsubscribeFromGuestStatistics (p. 501) to stop receiving statistics events. A callback function (event handler) implementation is required to use subscriptions.

The PHT_SYSTEM_STATISTICS object provides methods for obtaining handles to objects containing the performance data of a specific type. It also provides methods for obtaining memory statistics directly from it. The steps in obtaining the performance data are as follows:

1. Obtain a handle of type PHT_SYSTEM_STATISTICS.
2. To obtain memory statistics, use the functions provided by the object.
3. For other statistics (CPU, disk, network, etc.), first use a corresponding function to obtain a handle of a particular type, and

then use the functions of the received object to get the statistics.

Functions

	Name	Description
	PrlStat_GetCpusStatsCount (p. 361)	Determines the number of handles of type PHT_SYSTEM_STATISTICS_CPU (p. 365) contained in the specified PHT_SYSTEM_STATISTICS object.
	PrlStat_GetCpuStat (p. 362)	Obtains a handle of type PHT_SYSTEM_STATISTICS_CPU (p. 365) containing the specified CPU statistics.
	PrlStat_GetFreeRamSize (p. 362)	Returns free RAM size in bytes.
	PrlStat_GetOsUptime (p. 363)	Returns the total virtual machine usage time in seconds.
	PrlStat_GetRealRamSize (p. 363)	Returns the amount of physical memory, in bytes, currently occupied by the virtual machine.
	PrlStat_GetTotalRamSize (p. 364)	Returns total RAM size in bytes.
	PrlStat_GetUsageRamSize (p. 364)	Returns the size RAM currently in use (in bytes).

PrlStat_GetCpusStatsCount Function

Determines the number of handles of type PHT_SYSTEM_STATISTICS_CPU (p. 365) contained in the specified PHT_SYSTEM_STATISTICS (p. 360) object.

Syntax

```
PRL_RESULT PrlStat_GetCpusStatsCount(
    PRL_HANDLE hStat,
    PRL_UINT32_PTR pnCount
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

pnCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Each available CPU in the system will have its own PHT_SYSTEM_STATISTICS_CPU (p. 365) object instance.

See Also

PrlStat_GetCpuStat (p. 362)

PrlStat_GetCpuStat Function

Obtains a handle of type PHT_SYSTEM_STATISTICS_CPU (p. 365) containing the specified CPU statistics.

Syntax

```
PRL_RESULT PrlStat_GetCpuStat(
    PRL_HANDLE hStat,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phCpuStat
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

nIndex

A numeric index identifying the CPU statistics object instance (beginning with 0). To get the number of instances, use the PrlStat_GetCpusStatsCount (p. 361) function.

phCpuStat

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStat_GetFreeRamSize Function

Returns free RAM size in bytes.

Syntax

```
PRL_RESULT PrlStat_GetFreeRamSize(
    PRL_HANDLE hStat,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

pnSize

[out] A pointer to a variable that receives the result (free RAM size in bytes).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStat_GetOsUptime Function

Returns the total virtual machine usage time in seconds.

Syntax

```
PRL_RESULT PrlStat_GetOsUptime(
    PRL_HANDLE hStat,
    PRL_UINT64_PTR pnUptime
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

pnUptime

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The virtual machine usage counter is not reset when the virtual machine is stopped and restarted, so it's value reflects the total uptime since the virtual machine was first started.

PrlStat_GetRealRamSize Function

Returns the amount of physical memory, in bytes, currently occupied by the virtual machine.

Syntax

```
PRL_RESULT PrlStat_GetRealRamSize(
    PRL_HANDLE hStat,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

pnSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStat_GetTotalRamSize Function

Returns total RAM size in bytes.

Syntax

```
PRL_RESULT PrlStat_GetTotalRamSize(
    PRL_HANDLE hStat,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

pnSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStat_GetUsageRamSize Function

Returns the size RAM currently in use (in bytes).

Syntax

```
PRL_RESULT PrlStat_GetUsageRamSize(
    PRL_HANDLE hStat,
    PRL_UINT64_PTR pnSize
);
```

File

PrlApiStat.h

Parameters

hStat

A handle of type PHT_SYSTEM_STATISTICS (p. 360).

pnSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_SYSTEM_STATISTICS_CPU

The PHT_SYSTEM_STATISTICS_CPU handle is used to obtain CPU statistics. The handle is obtained by making the PrlStat_GetCpuStat (p. 362) function call.

Functions

	Name	Description
	PrlStatCpu_GetCpuUsage (p. 365)	Returns the CPU usage, in percents.
	PrlStatCpu_GetSystemTime (p. 366)	Returns the CPU system time, in seconds.
	PrlStatCpu_GetTotalTime (p. 366)	Returns the CPU total time in seconds.
	PrlStatCpu_GetUserTime (p. 367)	Returns the CPU user time in seconds.

PrlStatCpu_GetCpuUsage Function

Returns the CPU usage, in percents.

Syntax

```
PRL_RESULT PrlStatCpu_GetCpuUsage(
    PRL_HANDLE hCpuStat,
    PRL_UINT32_PTR pnPercents
);
```

File

PrlApiStat.h

Parameters

hCpuStat

A handle of type PHT_SYSTEM_STATISTICS_CPU (p. 365) identifying the CPU.

pnPercents

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStatCpu_GetSystemTime Function

Returns the CPU system time, in seconds.

Syntax

```
PRL_RESULT PrlStatCpu_GetSystemTime(
    PRL_HANDLE hCpuStat,
    PRL_UINT64_PTR pnTime
);
```

File

PrlApiStat.h

Parameters

hCpuStat

A handle of type PHT_SYSTEM_STATISTICS_CPU (p. 365) identifying the CPU.

pnTime

[out] A pointer to a variable that receives the

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStatCpu_GetTotalTime Function

Returns the CPU total time in seconds.

Syntax

```
PRL_RESULT PrlStatCpu_GetTotalTime(
    PRL_HANDLE hCpuStat,
    PRL_UINT64_PTR pnTime
);
```

File

PrlApiStat.h

Parameters

hCpuStat

A handle of type PHT_SYSTEM_STATISTICS_CPU (p. 365) identifying the CPU.

pnTime

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlStatCpu_GetUserTime Function

Returns the CPU user time in seconds.

Syntax

```
PRL_RESULT PrlStatCpu_GetUserTime(
    PRL_HANDLE hCpuStat,
    PRL_UINT64_PTR pnTime
);
```

File

PrlApiStat.h

Parameters

hCpuStat

A handle of type PHT_SYSTEM_STATISTICS_CPU (p. 365) identifying the CPU.

pnTime

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_SYSTEM_STATISTICS_DISK

The PHT_SYSTEM_STATISTICS_DISK object is used to obtain disk statistics. The handle is obtained by making the PrlStat_GetDiskStat function call.

PHT_SYSTEM_STATISTICS_DISK_PARTITION

The PHT_SYSTEM_STATISTICS_DISK_PARTITION handle is used to obtain disk partition statistics. The handle is obtained by making the PrlStatDisk_GetPartStat function call.

PHT_SYSTEM_STATISTICS_IFACE

The PHT_SYSTEM_STATISTICS_IFACE handle is used to obtain network interface statistics. The handle is obtained by making the PrlStat_GetIfaceStat function call.

PHT_SYSTEM_STATISTICS_PROCESS

The PHT_SYSTEM_STATISTICS_PROCESS handle is used to obtain system process statistics. The handle is obtained by making the PrlStat_GetProcStat function call.

PHT_SYSTEM_STATISTICS_USER_SESSION

The PHT_SYSTEM_STATISTICS_USER_SESSION handle is used to obtain user session statistics. The handle is obtained by making the PrlStat_GetUserStat function call.

PHT_SYSTEM_STATISTICS_VM_DATA

PHT_UIEMU_INPUT

PHT_USB_IDENTITY

Functions

	Name	Description
	PrlUsbIdent_GetFriendlyName (p. 369)	Returns user friendly name of USB device.
	PrlUsbIdent_GetSystemName (p. 369)	Returns system name for USB device.
	PrlUsbIdent_GetVmUuidAssociation (p. 370)	Returns Uuid of Vm associated with USB device or empty string if no association is specified.

PrlUsbIdent_GetFriendlyName Function

Returns user friendly name of USB device.

Syntax

```
PRL_RESULT PrlUsbIdent_GetFriendlyName(
    PRL_HANDLE hUsbIdent,
    PRL_STR sFriendlyName,
    PRL_UINT32_PTR pnFriendlyNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hUsbIdent

A handle of type PHT_USB_IDENTITY (p. 368).

sFriendlyName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnFriedlyNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsbIdent_GetSystemName Function

Returns system name for USB device.

Syntax

```
PRL_RESULT PrlUsbIdent_GetSystemName(
    PRL_HANDLE hUsbIdent,
    PRL_STR sSystemName,
    PRL_UINT32_PTR pnSystemNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hUsbIdent

A handle of type PHT_USB_IDENTITY (p. 368).

sSystemName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnSystemNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsbIdent_GetVmUuidAssociation Function

Returns Uuid of Vm associated with USB device or empty string if no association is specified.

Syntax

```
PRL_RESULT PrlUsbIdent_GetVmUuidAssociation(
    PRL_HANDLE hUsbIdent,
    PRL_STR sVmUuid,
    PRL_UINT32_PTR pnVmUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hUsbIdent

A handle of type PHT_USB_IDENTITY (p. 368).

sVmUuid

[out] A pointer to a buffer that receives the result (null-terminated string).

pnVmUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_USER_INFO

The PHT_USER_INFO handle is a container that contains information about an individual user. The handle is obtained by making the PrlSrv_GetUserInfo (p. 260) or the PrlSrv_GetUserInfoList (p. 261) function call. The handle can be obtained for any user (not just the currently logged in user). To obtain the information for the currently logged in user, see the PHT_USER_PROFILE (p. 374) handle.

Functions

	Name	Description
	PrlUsrInfo_CanChangeSrvSets (p. 371)	Determines whether the specified user is allowed to modify Virtuozzo Service preferences.
	PrlUsrInfo_GetDefaultVmFolder (p. 372)	Returns name and path of the default virtual machine directory for the specified user.
	PrlUsrInfo_GetName (p. 372)	Returns the user name.
	PrlUsrInfo_GetSessionCount (p. 373)	Returns the user active session count.
	PrlUsrInfo_GetUuid (p. 374)	Returns the user Universally Unique Identifier (UUID).

PrlUsrInfo_CanChangeSrvSets Function

Determines whether the specified user is allowed to modify Virtuozzo Service preferences.

Syntax

```
PRL_RESULT PrlUsrInfo_CanChangeSrvSets(
    PRL_HANDLE hUserInfo,
    PRL_BOOL_PTR pbCanChangeSrvSets
);
```

File

PrlApiDisp.h

Parameters

hUserInfo

A handle of type PHT_USER_INFO (p. 371) identifying the user.

pbCanChangeSrvSets

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the user is allowed to modify Virtuozzo Service preferences. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsrInfo_GetDefaultVmFolder Function

Returns name and path of the default virtual machine directory for the specified user.

Syntax

```
PRL_RESULT PrlUsrInfo_GetDefaultVmFolder(
    PRL_HANDLE hUserInfo,
    PRL_STR sDefaultVmFolder,
    PRL_UINT32_PTR pnDefaultVmFolderBufLength
);
```

File

PrlApiDisp.h

Parameters

hUserInfo

A handle of type PHT_USER_INFO (p. 371) identifying the user.

sDefaultVmFolder

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDefaultVmFolderBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If the user doesn't have a default virtual machine directory, the function will return the name and path of the default directory for this Virtuozzo Service. To set a default virtual machine directory for a user, use the PrlUsrCfg_SetDefaultVmFolder (p. 378) function.

See Also

PrlUsrCfg_SetDefaultVmFolder (p. 378)

PrlUsrInfo_GetName Function

Returns the user name.

Syntax

```
PRL_RESULT PrlUsrInfo_GetName(
    PRL_HANDLE hUserInfo,
    PRL_STR sName,
```

```
    PRL_UINT32_PTR pnNameBufLength
);
```

File

PrlApiDisp.h

Parameters

hUserInfo

A handle of type PHT_USER_INFO (p. 371) identifying the user.

sName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsrInfo_GetSessionCount Function

Returns the user active session count.

Syntax

```
PRL_RESULT PrlUsrInfo_GetSessionCount(
    PRL_HANDLE hUserInfo,
    PRL_UINT32_PTR pnSessionCount
);
```

File

PrlApiDisp.h

Parameters

hUserInfo

A handle of type PHT_USER_INFO (p. 371) identifying the user.

pnSessionCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsrInfo_GetUuid Function

Returns the user Universally Unique Identifier (UUID).

Syntax

```
PRL_RESULT PrlUsrInfo_GetUuid(
    PRL_HANDLE hUserInfo,
    PRL_STR sUuid,
    PRL_UINT32_PTR pnUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hUserInfo

A handle of type PHT_USER_INFO (p. 371) identifying the user.

sUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_USER_PROFILE

The PHT_USER_PROFILE handle is used to obtain information about the currently logged in user and for setting the user preferences. The handle is obtained by calling the PrlSrv_GetUserProfile (p. 262) function. To obtain information about any user (not just the currently logged on user), see the PHT_USER_INFO (p. 371) handle.

Description

All modifications to the user profile must begin with the PrlSrv_UserProfileBeginEdit (p. 313) function call and end with the PrlSrv_UserProfileCommit (p. 314) call. The functions are used to detect collisions with other clients trying to modify the profile of the same user at the same time. The functions do not lock the profile being edited. If a collision is detected on commit, the changes are rejected.

Functions

	Name	Description
	PrlUsrCfg_CanChangeSrvSets (p. 375)	Determines if the current user can modify Virtuozzo Service preferences (PHT_DISP_CONFIG (p. 32)).
	PrlUsrCfg_CanUseMngConsole (p. 376)	Determines if the user is allowed to use the Virtuozzo Service management console utility.
	PrlUsrCfg_GetDefaultVmFolder (p. 376)	Returns name and path of the default virtual machine directory for this user.
	PrlUsrCfg_GetVmDirUuid (p. 377)	Returns name and path of the default virtual machine folder for this Virtuozzo Service.
	PrlUsrCfg_IsLocalAdministrator (p. 378)	Determines if the user is a local administrator account on the host where Virtuozzo Service started.
	PrlUsrCfg_SetDefaultVmFolder (p. 378)	Allows to set a default virtual machine folder for the user.
	PrlUsrCfg_SetVmDirUuid (p. 379)	Sets user default VM dir.

PrlUsrCfg_CanChangeSrvSets Function

Determines if the current user can modify Virtuozzo Service preferences (PHT_DISP_CONFIG (p. 32)).

Syntax

```
PRL_RESULT PrlUsrCfg_CanChangeSrvSets(
    PRL_HANDLE hUserProfile,
    PRL_BOOL_PTR pbCanChangeSrvSets
);
```

File

PrlApiDisp.h

Parameters

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

pbCanChangeSrvSets

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the client can modify Virtuozzo Service preferences. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

By default, only administrators of the host machine can modify Virtuozzo Service preferences. Depending on the current Virtuozzo Service settings, other users may also be granted this privilege at the time a user profile is created. To verify the default settings, use the PrlDispCfg_CanChangeDefaultSettings (p. 36) function. Whether the user can or cannot modify Virtuozzo Service preferences, this setting is final and cannot itself be changed through Virtuozzo API.

See Also

PrlDispCfg_CanChangeDefaultSettings (p. 36)

PrlDispCfg_SetCanChangeDefaultSettings (p. 59)

PrlUsrCfg_CanUseMngConsole Function

Determines if the user is allowed to use the Virtuozzo Service management console utility.

Syntax

```
PRL_RESULT PrlUsrCfg_CanUseMngConsole(
    PRL_HANDLE hUserProfile,
    PRL_BOOL_PTR pbCanUseMngConsole
);
```

File

PrlApiDisp.h

Parameters

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

pbCanUseMngConsole

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the user is allows to use the utility. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsrCfg_GetDefaultVmFolder Function

Returns name and path of the default virtual machine directory for this user.

Syntax

```
PRL_RESULT PrlUsrCfg_GetDefaultVmFolder(
    PRL_HANDLE hUserProfile,
    PRL_STR sDefaultVmFolder,
    PRL_UINT32_PTR pnDefaultVmFolderBufLength
);
```

File

PrlApiDisp.h

Parameters

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

sDefaultVmFolder

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDefaultVmFolderBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If the user doesn't have a default virtual machine folder (if it was not previously set up by the PrlUsrCfg_SetDefaultVmFolder (p. 378) function call), the function returns an empty string. In such a case, use the PrlUsrCfg_GetVmDirUuid (p. 377) function to get the virtual machine folder for the Virtuozzo Service, which is used by default for all users.

See Also

PrlUsrCfg_SetDefaultVmFolder (p. 378)

PrlUsrCfg_GetVmDirUuid Function

Returns name and path of the default virtual machine folder for this Virtuozzo Service.

Syntax

```
PRL_RESULT PrlUsrCfg_GetVmDirUuid(
    PRL_HANDLE hUserProfile,
    PRL_STR sVmDirUuid,
    PRL_UINT32_PTR pnVmDirUuidBufLength
);
```

File

PrlApiDisp.h

Parameters

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

sVmDirUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmDirUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If a default virtual machine folder has been set up for this user with the PrlUsrCfg_SetDefaultVmFolder (p. 378) function, use the PrlUsrCfg_GetDefaultVmFolder (p. 376) to get its name and path.

PrlUsrCfg_IsLocalAdministrator Function

Determines if the user is a local administrator account on the host where Virtuozzo Service started.

Syntax

```
PRL_RESULT PrlUsrCfg_IsLocalAdministrator(
    PRL_HANDLE hUserProfile,
    PRL_BOOL_PTR pbIsLocalAdmin
);
```

File

PrlApiDisp.h

Parameters

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

pbIsLocalAdmin

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the user is local administrator. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlUsrCfg_SetDefaultVmFolder Function

Allows to set a default virtual machine folder for the user.

Syntax

```
PRL_RESULT PrlUsrCfg_SetDefaultVmFolder(
    PRL_HANDLE hUserProfile,
    PRL_CONST_STR sNewDefaultVmFolder
);
```

File

PrlApiDisp.h

Parameters

hUserProfile

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

sNewDefaultVmFolder

Name and path of the folder to set as default virtual machine folder for the user. The folder must already exist on the host.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A Virtuozzo Service has a default virtual machine directory (the directory where new virtual machines are created). Individual users can have their own default virtual machine directories if needed. If a user has its own virtual machine directory set up, it takes precedence over the Virtuozzo Service default. To set a default directory for a user, first call the PrlSrv_UserProfileBeginEdit (p. 313) function to mark the beginning of the operation. Then call the PrlUsrCfg_SetDefaultVmFolder function to set the new default virtual machine folder. Finally, call the PrlSrv_UserProfileCommit (p. 314) function to save your changes.

PrlUsrCfg_SetVmDirUuid Function

Sets user default VM dir.

Syntax

```
PRL_RESULT PrlUsrCfg_SetVmDirUuid(
    PRL_HANDLE hUserProfile,
    PRL_CONST_STR sNewVmDirUuid
);
```

File

PrlApiDisp.h

Parameters

handle

A handle of type PHT_USER_PROFILE (p. 374) identifying the user.

unnamed_param

new VM dir

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_USER_PROFILE (p. 374) handle or non valid (NULL) pointer to the new VM dir string was specified PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

Remarks

Specified string value must be UTF-8 encoding and ends with '0' terminating symbol.

PHT_VCMMD_CONFIG

Functions

	Name	Description
	PrlVcmmdConfig_GetPolicy (p. 380)	Obtains vcmmd policy.

	PrlVcmmdConfig_SetPolicy (p. 380)	Set vcmmd config value.

PrlVcmmdConfig_GetPolicy Function

Obtains vcmmd policy.

Syntax

```
PRL_RESULT PrlVcmmdConfig_GetPolicy(
    PRL_HANDLE hVcmmdConfig,
    PRL_STR sPolicy,
    PRL_UINT32_PTR pnSize
);
```

File

PrlApiDisp.h

Remarks

- Parameters
- hVcmmdConfig : A handle of type PHT_VCMMD_CONFIG (p. 379).
- sPolicy : [out] Pointer to a buffer that receives
- vcmmd policy (a UTF-8 encoded,
- null-terminated string).
- pnSize : [in] The size of the output buffer (in
- bytes). Set the buffer pointer to null
- and this value to zero to receive the
- required size.
- [out] The required output buffer size.
- nFlags : reserved parameter
- Returns
- PRL_RESULT. Possible values are:
- PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

*

- PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVcmmdConfig_SetPolicy Function

Set vcmmd config value.

Syntax

```
PRL_RESULT PrlVcmmdConfig_SetPolicy(
    PRL_HANDLE hVcmmdConfig,
    PRL_CONST_STR sPolicy
);
```

File

PrlApiDisp.h

Remarks

- Parameters
- hVcmmdConfig : A handle of type PHT_VCMMD_CONFIG (p. 379).
- sPolicy : [in] vcmmd policy to set
- Returns
- PRL_RESULT. Possible values are:
- PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.
- *
- PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_VIDEO_RECEIVER

PHT_VIRTUAL_DEV_FLOPPY

The PHT_VIRTUAL_DEV_FLOPPY handle contains information about a virtual floppy disk in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetFloppyDisk (p. 575) function call.

See Also

Common Virtual Device Functions (p. 752)

PHT_VIRTUAL_DEV_HARD_DISK

The PHT_VIRTUAL_DEV_HARD_DISK handle is used to manage virtual hard disks in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetHardDisk (p. 580) function call.

Functions

	Name	Description
	PrlVmDevHd_AddPartition (p. 382)	Assigns a boot camp partition to the specified virtual hard disk.
	PrlVmDevHd_GetDiskSize (p. 383)	Returns the hard disk size.
	PrlVmDevHd_GetDiskType (p. 384)	Returns the hard disk type.
	PrlVmDevHd_GetEncryption (p. 384)	Returns the disk encryption parameters
	PrlVmDevHd_GetMountPoint (p. 385)	Returns the mount point of the specified virtual hard disk.

	PrlVmDevHd_GetPartition (p. 385)	Obtains a handle of type PHT_VIRTUAL_DEV_HD_PARTITION (p. 397) containing information about the specified partition of the virtual hard disk.
	PrlVmDevHd_GetPartitionsCount (p. 386)	Determines the number of partitions on the virtual hard disk.
	PrlVmDevHd_GetSerialNumber (p. 387)	Return disk serial number.
	PrlVmDevHd_GetSizeOnDisk (p. 387)	Returns the size of the occupied space on the hard disk.
	PrlVmDevHd_GetStorageURL (p. 388)	Return custom storage URL.
	PrlVmDevHd_IsAutoCompressEnabled (p. 389)	Returns the auto compress flag of the specified virtual hard disk.
	PrlVmDevHd_IsSplitted (p. 389)	Determines if the virtual hard disk is split into multiple files.
	PrlVmDevHd_SetAutoCompressEnabled (p. 390)	Sets the auto compress sign flag of the specified virtual hard disk.
	PrlVmDevHd_SetDiskSize (p. 391)	Sets the size of the specified virtual hard disk.
	PrlVmDevHd_SetDiskType (p. 391)	Sets the type of the specified virtual hard disk.
	PrlVmDevHd_SetEncryption (p. 392)	Specify disk encryption parameters
	PrlVmDevHd_SetMountPoint (p. 392)	Sets the mount point of the specified virtual hard disk.
	PrlVmDevHd_SetSerialNumber (p. 393)	Set disk serial number.
	PrlVmDevHd_SetSplitted (p. 394)	Allows to specify whether the hard disk should be split into multiple files or if a single file should be used.
	PrlVmDevHd_SetStorageURL (p. 394)	Set custom storage URL.
	PrlVmDevNet_GetHostInterfaceName (p. 395)	Obtains the custom host adapter name.
	PrlVmDevNet_GetMacAddressCanonical (p. 396)	Returns the MAC address of a virtual network adapter inside a virtual machine (in human-friendly format xx:xx:xx:xx:xx:xx).
	PrlVmDevNet_SetHostInterfaceName (p. 396)	Sets the custom host adapter name in a bridged networking mode, (autogenerated if not set) hVmDev : A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

See Also

Common Virtual Device Functions (p. 752)

PrlVmDevHd_AddPartition Function

Assigns a boot camp partition to the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_AddPartition(
    PRL_HANDLE hVmDev,
    PRL_HANDLE_PTR phPartition
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

phPartition

[out] A pointer to a variable that receives a handle of type PHT_VIRTUAL_DEV_HD_PARTITION (p. 397) containing the new partition information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When creating a virtual machine based on a boot camp partition, this function must be used to obtain a handle of type PHT_VIRTUAL_DEV_HD_PARTITION (p. 397). The handle can then be used to set the system name of the boot camp partition that you would like to use.

PrlVmDevHd_GetDiskSize Function

Returns the hard disk size.

Syntax

```
PRL_RESULT PrlVmDevHd_GetDiskSize(
    PRL_HANDLE hVmDev,
    PRL_UINT32_PTR pnDiskSize
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

pnDiskSize

[out] Pointer to a variable that receives the hard disk size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_SetDiskSize (p. 391)

PrlVmDevHd_GetDiskType Function

Returns the hard disk type.

Syntax

```
PRL_RESULT PrlVmDevHd_GetDiskType(
    PRL_HANDLE hVmDev,
    PRL_HARD_DISK_INTERNAL_FORMAT_PTR pnDiskType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

pnDiskType

[out] Pointer to a variable that receives the disk type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A virtual hard disk can be one of two types: expanding or fixed. The size of an expanding hard disk is expanded dynamically up to a predefined limit. The size of a fixed hard disk is set to the specified value at the time of creation.

See Also

PrlVmDevHd_SetDiskType (p. 391)

PrlVmDevHd_GetEncryption Function

Returns the disk encryption parameters

Syntax

```
PRL_RESULT PrlVmDevHd_GetEncryption(
    PRL_HANDLE hVmDev,
    PRL_HANDLE_PTR phEncryption
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

phEncryption

[out] A pointer to a variable that receives the PHT_DISK_ENCRYPTION handle containing the disk encryption parameters.

Returns

PRL_RESULT.

PrlVmDevHd_GetMountPoint Function

Returns the mount point of the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_GetMountPoint(
    PRL_HANDLE hVmDev,
    PRL_STR sMountPoint,
    PRL_UINT32_PTR pnMountPointBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

sMountPoint

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnMountPointBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_SetMountPoint (p. 392),

PrlVmDevHd_GetPartition Function

Obtains a handle of type PHT_VIRTUAL_DEV_HD_PARTITION (p. 397) containing information about the specified partition of the virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_GetPartition(
```

```
    PRL_HANDLE hVmDev,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phPartition
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

nIndex

An index identifying the partition instance (beginning with 0). To get the number of partitions of the virtual hard disk, use the PrlVmDevHd_GetPartitionsCount (p. 386) function.

phPartition

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_GetPartitionsCount (p. 386)

PrlVmDevHd_GetPartitionsCount Function

Determines the number of partitions on the virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_GetPartitionsCount(
    PRL_HANDLE hVmDev,
    PRL_UINT32_PTR pnPartitionsCount
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

pnPartitionsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_GetPartition (p. 385)

PrlVmDevHd_GetSerialNumber Function

Return disk serial number.

Syntax

```
PRL_RESULT PrlVmDevHd_GetSerialNumber(
    PRL_HANDLE hVmDev,
    PRL_STR sSerialNumber,
    PRL_UINT32_PTR pnSerialNumberBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

sSerialNumber

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnSerialNumberBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevHd_GetSizeOnDisk Function

Returns the size of the occupied space on the hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_GetSizeOnDisk(
    PRL_HANDLE hVmDev,
    PRL_UINT32_PTR pnOnDiskSize
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

pnOnDiskSize

[out] Pointer to a variable that receives the size. The size is specified in bytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevHd_GetStorageURL Function

Return custom storage URL.

Syntax

```
PRL_RESULT PrlVmDevHd_GetStorageURL(
    PRL_HANDLE hVmDev,
    PRL_STR sURL,
    PRL_UINT32_PTR pnURLBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

sURL

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnURLBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_SetStorageURL (p. 394)

PrlVmDevHd_IsAutoCompressEnabled Function

Returns the auto compress flag of the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_IsAutoCompressEnabled(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

pbEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

(for Virtuozzo Container only)

See Also

PrlVmDevHd_IsAutoCompressEnabled,

PrlVmDevHd_IsSplitted Function

Determines if the virtual hard disk is split into multiple files.

Syntax

```
PRL_RESULT PrlVmDevHd_IsSplitted(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbSplitted
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

pbSplitted

[out] Pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the disk is split into multiple files. PRL_FALSE (p. 1626) indicates that a single file is used.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When adding a virtual hard disk to a virtual machine, you have an option of splitting the disk into multiple files. By using this function, you can determine whether this options was used or not.

See Also

PrlVmDevHd_SetSplitted (p. 394)

PrlVmDevHd_SetAutoCompressEnabled Function

Sets the auto compress sign flag of the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_SetAutoCompressEnabled(
    PRL_HANDLE hVmDev,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

bEnabled

Specify PRL_FALSE (p. 1626) to disable auto compress

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

(for Virtuozzo Container only)

See Also

PrlVmDevHd_GetMountPoint (p. 385)

PrlVmDevHd_SetDiskSize Function

Sets the size of the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_SetDiskSize(
    PRL_HANDLE hVmDev,
    PRL_UINT32 nDiskSize
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

nDiskSize

The new size in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when adding a new virtual hard disk or when modifying the size of an existing disk.

See Also

PrlVmDevHd_GetDiskSize (p. 383)

PrlVmDevHd_SetDiskType Function

Sets the type of the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_SetDiskType(
    PRL_HANDLE hVmDev,
    PRL_HARD_DISK_INTERNAL_FORMAT nDiskType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

nDiskType

The hard disk type to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A virtual hard disk can be one of the two types: expanding or fixed-size. The size of an expanding hard disk is expanded dynamically up to a predefined limit. The size of a fixed-size hard disk is set to the specified value at the time of creation. The function can be used with both new and existing virtual hard disks.

See Also

PrlVmDevHd_GetDiskType (p. 384)

PrlVmDevHd_SetEncryption Function

Specify disk encryption parameters

Syntax

```
PRL_RESULT PrlVmDevHd_SetEncryption(
    PRL_HANDLE hVmDev,
    PRL_HANDLE hEncryption
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

phEncryption

PHT_DISK_ENCRYPTION handle containing the disk encryption parameters.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevHd_SetMountPoint Function

Sets the mount point of the specified virtual hard disk.

Syntax

```
PRL_RESULT PrlVmDevHd_SetMountPoint(
    PRL_HANDLE hVmDev,
```

```
    PRL_CONST_STR sMountPoint
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

sMountPoint

Mount point path

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

(for Virtuozzo Container only)

See Also

PrlVmDevHd_GetMountPoint (p. 385)

PrlVmDevHd_SetSerialNumber Function

Set disk serial number.

Syntax

```
PRL_RESULT PrlVmDevHd_SetSerialNumber(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sSerialNumber
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

sSerialNumber

[in] Serial number - a UTF-8 encoded string with null '0' symbol at the end.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_GetSerialNumber (p. 387)

PrlVmDevHd_SetSplitted Function

Allows to specify whether the hard disk should be split into multiple files or if a single file should be used.

Syntax

```
PRL_RESULT PrlVmDevHd_SetSplitted(
    PRL_HANDLE hVmDev,
    PRL_BOOL bSplitted
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

bSplitted

Hard disk file option. Specify PRL_TRUE (p. 1652) to split the disk into multiple files. Specify PRL_FALSE (p. 1626) to use a single file.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHd_IsSplitted (p. 389)

PrlVmDevHd_SetStorageURL Function

Set custom storage URL.

Syntax

```
PRL_RESULT PrlVmDevHd_SetStorageURL(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sURL
);
```

File

PrlApiVm.h

Parameters

hVmDev

A valid handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) identifying the virtual hard disk.

sURL

[in] Storage URL - an UTF-8 encoded string with null '0' symbol at the end. The following URL types are supported: "backup://backup_server/backup_UUID/disk_number" - use disk backup, identified by "backup_UUID/disk_number" from server "backup_server".

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A storage URL is required when actual data of a virtual hard disk is accessed using some external tool or protocol. For example, a VM disk backup, which resides on a remote server, could be attached as a data source using the following storage URL: "backup://backup_server/backup_UUID/disk_number". Here, instead of reading data from the locally stored disk image, a VM will connect to the "backup_server" and read data from the "backup_UUID/disk_number" backup.

See Also

PrlVmDevHd_GetStorageURL (p. 388)

PrlVmDevNet_GetHostInterfaceName Function

Obtains the custom host adapter name.

Syntax

```
PRL_RESULT PrlVmDevNet_GetHostInterfaceName(
    PRL_HANDLE hVmDev,
    PRL_STR sHostInterfaceName,
    PRL_UINT32_PTR pnHostInterfaceNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sHostInterfaceName

[out] A pointer to a buffer that receives the name (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_GetMacAddressCanonical Function

Returns the MAC address of a virtual network adapter inside a virtual machine (in human-friendly format xx:xx:xx:xx:xx:xx).

Syntax

```
PRL_RESULT PrlVmDevNet_GetMacAddressCanonical(
    PRL_HANDLE hVmDev,
    PRL_STR sMacAddress,
    PRL_UINT32_PTR pnMacAddressBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sMacAddress

[out] Pointer to a buffer that receives the MAC address (a UTF-8 encoded, null-terminated string).

pnMacAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetMacAddress (p. 420)

PrlVmDevNet_SetHostInterfaceName Function

Sets the custom host adapter name in a bridged networking mode, (autogenerated if not set) hVmDev : A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

Syntax

```
PRL_RESULT PrlVmDevNet_SetHostInterfaceName(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewHostInterfaceName
);
```

File

PrlApiVm.h

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

sNewHostInterfaceName : Host interface name.

PHT_VIRTUAL_DEV_HD_PARTITION

The PHT_VIRTUAL_DEV_HD_PARTITION handle is used to manage partitions of a virtual hard disk in a virtual machine. The handle is obtained by making the PrlVmDevHd_AddPartition (p. 382) or the PrlVmDevHd_GetPartition (p. 385) function call.

Functions

	Name	Description
	PrlVmDevHdPart_GetSysName (p. 397)	Returns the system name of the specified disk partition.
	PrlVmDevHdPart_Remove (p. 398)	Removes the specified partition handle from the virtual hard disk list.
	PrlVmDevHdPart_SetSysName (p. 398)	Sets system name for the specified disk partition.

PrlVmDevHdPart_GetSysName Function

Returns the system name of the specified disk partition.

Syntax

```
PRL_RESULT PrlVmDevHdPart_GetSysName(
    PRL_HANDLE hPartition,
    PRL_STR sSysName,
    PRL_UINT32_PTR pnSysNameBufLength
);
```

File

PrlApiVm.h

Parameters

hPartition

A handle of type PHT_VIRTUAL_DEV_HD_PARTITION (p. 397) identifying the disk partition.

sSysName

[out] A pointer to a buffer that receives the name (a UTF-8 encoded, null-terminated string).

pnSysNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHdPart_SetSysName (p. 398)

PrlVmDevHdPart_Remove Function

Removes the specified partition handle from the virtual hard disk list.

Syntax

```
PRL_RESULT PrlVmDevHdPart_Remove(
    PRL_HANDLE hPartition
);
```

File

PrlApiVm.h

Parameters

hPartition

A handle identifying the partition.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevHdPart_SetSysName Function

Sets system name for the specified disk partition.

Syntax

```
PRL_RESULT PrlVmDevHdPart_SetSysName(
    PRL_HANDLE hPartition,
    PRL_CONST_STR sSysName
);
```

File

PrlApiVm.h

Parameters

hPartition

A handle of type PHT_VIRTUAL_DEV_HD_PARTITION (p. 397) identifying the disk partition.

sSysName

A system name to set for the partition.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHdPart_GetSysName (p. 397)

PHT_VIRTUAL_DEV_NET_ADAPTER

The PHT_VIRTUAL_DEV_NET_ADAPTER handle is used to manage network adapters in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetNetAdapter (p. 589) function call.

Functions

	Name	Description
	PrlVmDevNet_GenerateMacAddr (p. 401)	Generates a unique MAC address for a virtual network adapter on a virtual machine.
	PrlVmDevNet_GetAdapterType (p. 401)	Obtain the adapter type assigned to a virtual network adapter.
	PrlVmDevNet_GetBoundAdapterIndex (p. 402)	Returns the index of the adapter to which the specified virtual adapter is bound.
	PrlVmDevNet_GetBoundAdapterName (p. 402)	Returns the name of the adapter to which the specified virtual adapter is bound.
	PrlVmDevNet_GetDefaultGateway (p. 403)	Obtains the default gateway assigned to a virtual network adapter in a virtual machine.
	PrlVmDevNet_GetDefaultGatewayIPv6 (p. 404)	Obtains the default gateway (IPv6) assigned to a virtual network adapter in a virtual machine.
	PrlVmDevNet_GetDnsServers (p. 405)	Obtains the list of DNS servers which is assigned to a virtual network adapter on virtual machine startup.
	PrlVmDevNet_GetFirewallDefaultPolicy (p. 405)	Gets an adapter firewall default policy.
	PrlVmDevNet_GetFirewallRuleList (p. 406)	Returns the firewall rule entries list.
	PrlVmDevNet_GetMacAddress (p. 406)	Returns the MAC address of a virtual network adapter inside a virtual machine (in Virtuozzo format XXXXXXXXXXXX).
	PrlVmDevNet_GetNetAddresses (p. 407)	Obtains a list of IP addresses in the address/subnet_mask format which is assigned to a virtual network adapter on virtual machine startup.
	PrlVmDevNet_GetSearchDomains (p. 408)	Obtains a lists of search domains which are assigned to a virtual network adapter on virtual machine startup.
	PrlVmDevNet_GetVirtualNetworkId (p. 408)	Obtains the virtual network ID assigned to a virtual network adapter in a virtual machine.

	PrlVmDevNet_IsAutoApply (p. 409)	Determines if the specified network adapter is configured to automatically apply network settings inside guest.
	PrlVmDevNet_IsConfigureWithDhcp (p. 410)	Determines if the specified network adapter is configured through DHCP on the guest OS side.
	PrlVmDevNet_IsConfigureWithDhcpIPv6 (p. 410)	Determines if the specified network adapter is configured through DHCP (IPv6) on the guest OS side.
	PrlVmDevNet_IsFirewallEnabled (p. 411)	Gets state switched on or off of the adapter firewall.
	PrlVmDevNet_IsPktFilterPreventIpSpoof (p. 411)	Determines if the specified network adapter is configured to filter net packages and send packages only from IP addresses in the adapter NetAdrresses list.
	PrlVmDevNet_IsPktFilterPreventMacSpoof (p. 412)	Determines if the specified network adapter is configured to filter net packages and send packages only from its own MAC address.
	PrlVmDevNet_SetAdapterType (p. 412)	Allows to manually assign adapter type to a network adapter.
	PrlVmDevNet_SetAutoApply (p. 413)	Allows to specify whether a network adapter in a virtual machine should be automatically configured.
	PrlVmDevNet_SetBoundAdapterIndex (p. 414)	Sets the index of the adapter to which the specified virtual adapter should be bound.
	PrlVmDevNet_SetBoundAdapterName (p. 414)	Sets the name of the network adapter to which a virtual network adapter inside a virtual machine will bind.
	PrlVmDevNet_SetConfigureWithDhcp (p. 415)	Allows to specify whether a network adapter in a virtual machine should be configured through DHCP or manually.
	PrlVmDevNet_SetConfigureWithDhcpIPv6 (p. 416)	Allows to specify whether a network adapter in a virtual machine should be configured through DHCP (IPv6) or manually.
	PrlVmDevNet_SetDefaultGateway (p. 416)	Allows to manually set the default gateway address for a network adapter in a virtual machine.
	PrlVmDevNet_SetDefaultGatewayIPv6 (p. 417)	Allows to manually set the default gateway address (IPv6) for a network adapter in a virtual machine.
	PrlVmDevNet_SetDnsServers (p. 418)	Allows to manually assign DNS servers to a network adapter in a virtual machine.
	PrlVmDevNet_SetFirewallDefaultPolicy (p. 419)	Sets an adapter firewall default policy.
	PrlVmDevNet_SetFirewallEnabled (p. 419)	Sets state switched on or off of the adapter firewall.
	PrlVmDevNet_SetFirewallRuleList (p. 420)	Sets the firewall rule entries list.
	PrlVmDevNet_SetMacAddress (p. 420)	Allows to manually set MAC address to a network adapter inside a virtual machine.
	PrlVmDevNet_SetNetAddresses (p. 421)	Allows to manually set IP addresses/Subnet masks to a network adapter in a virtual machine.
	PrlVmDevNet_SetPktFilterPreventIpSpoof (p. 422)	Allows to specify whether a network adapter in a virtual machine should perform the filtration by IP addresses.
	PrlVmDevNet_SetPktFilterPreventMacSpoof (p. 422)	Allows to specify whether a network adapter in a virtual machine should filter packets through MAC address spoofing.
	PrlVmDevNet_SetPktFilterPreventPromisc (p. 423)	Allows to specify whether a network adapter in a virtual machine should receive packets sent not to VM.
	PrlVmDevNet_SetSearchDomains (p. 423)	Allows to manually assign search domains to a network adapter in a virtual machine.
	PrlVmDevNet_SetVirtualNetworkId (p. 424)	Allows to set the virtual network ID for a network adapter in a virtual machine.

See Also

Common Virtual Device Functions (p. 752)

PrlVmDevNet_GenerateMacAddr Function

Generates a unique MAC address for a virtual network adapter on a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_GenerateMacAddr(
    PRL_HANDLE hVmDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the network adapter.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The function will generate a new MAC address and will automatically assign it to the adapter. The function does not return the address to the caller. To retrieve the address, use the PrlVmDevNet_GetMacAddress (p. 406) function.

PrlVmDevNet_GetAdapterType Function

Obtain the adapter type assigned to a virtual network adapter.

Syntax

```
PRL_RESULT PrlVmDevNet_GetAdapterType(
    PRL_HANDLE hVmDev,
    PRL_VM_NET_ADAPTER_TYPE_PTR pnAdapterType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pnAdapterType

[out] A pointer to a buffer that receives the adapter type

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetAdapterType (p. 412)

PrlVmDevNet_GetBoundAdapterIndex Function

Returns the index of the adapter to which the specified virtual adapter is bound.

Syntax

```
PRL_RESULT PrlVmDevNet_GetBoundAdapterIndex(
    PRL_HANDLE hVmDev,
    PRL_INT32_PTR pnIndex
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pnIndex

[out] A pointer to a variable that receives the index. A returned value of -1 (negative one) indicates that the virtual network adapter is bound to the default adapter on the host.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

In a bridged networking mode, a network adapter inside a virtual machine is bound to an adapter on the host. This function allows to retrieve the index of the adapter from the host.

See Also

PrlVmDevNet_SetBoundAdapterIndex (p. 414)

PrlVmDevNet_GetBoundAdapterName Function

Returns the name of the adapter to which the specified virtual adapter is bound.

Syntax

```
PRL_RESULT PrlVmDevNet_GetBoundAdapterName(
    PRL_HANDLE hVmDev,
    PRL_STR sBoundAdapterName,
```

```
    PRL_UINT32_PTR pnBoundAdapterNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sBoundAdapterName

[out] Pointer to a buffer that receives the physical adapter name (a UTF-8 encoded, null-terminated string).

pnBoundAdapterNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

In a bridged networking mode, a network adapter inside a virtual machine is bound to an adapter on the host. This function allows to retrieve the name of the adapter from the host.

See Also

PrlVmDevNet_SetBoundAdapterName (p. 414)

PrlVmDevNet_GetDefaultGateway Function

Obtains the default gateway assigned to a virtual network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_GetDefaultGateway(
    PRL_HANDLE hVmDev,
    PRL_STR sDefaultGateway,
    PRL_UINT32_PTR pnDefaultGatewayBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sDefaultGateway

[out] A pointer to a buffer that receives the default gateway address (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetDefaultGateway (p. 416)

PrlVmDevNet_GetDefaultGatewayIPv6 Function

Obtains the default gateway (IPv6) assigned to a virtual network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_GetDefaultGatewayIPv6(
    PRL_HANDLE hVmDev,
    PRL_STR sDefaultGateway,
    PRL_UINT32_PTR pnDefaultGatewayBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sDefaultGateway

[out] A pointer to a buffer that receives the default gateway address (a UTF-8 encoded, null-terminated string).

pnDefaultGatewayBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetDefaultGateway (p. 416)

PrlVmDevNet_GetDnsServers Function

Obtains the list of DNS servers which is assigned to a virtual network adapter on virtual machine startup.

Syntax

```
PRL_RESULT PrlVmDevNet_GetDnsServers(
    PRL_HANDLE hVmDev,
    PRL_HANDLE_PTR phDnsServersList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

phDnsServersList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle that receives the list of DNS servers.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate strings list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetDnsServers (p. 418)

PrlVmDevNet_GetFirewallDefaultPolicy Function

Gets an adapter firewall default policy.

Syntax

```
PRL_RESULT PrlVmDevNet_GetFirewallDefaultPolicy(
    PRL_HANDLE hVmDev,
    PRL_FIREWALL_DIRECTION nDirection,
    PRL_FIREWALL_POLICY_PTR pnPolicy
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

nDirection

Direction of a firewall - to/from the virtual network adapter

pnPolicy

[out] A pointer to a buffer that receives the adapter firewall default policy.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Policy values are defined in the PRL_FIREWALL_POLICY (p. 1060) enumeration.

PrlVmDevNet_GetFirewallRuleList Function

Returns the firewall rule entries list.

Syntax

```
PRL_RESULT PrlVmDevNet_GetFirewallRuleList(
    PRL_HANDLE hVmDev,
    PRL_FIREWALL_DIRECTION nDirection,
    PRL_HANDLE_PTR phRuleList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

nDirection

Direction of a firewall - to/from the virtual network adapter

phRuleList

[out] A pointer to a variable that receives the new handle of PHT_HANDLES_LIST (p. 108) type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list or a port forward handle object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_GetMacAddress Function

Returns the MAC address of a virtual network adapter inside a virtual machine (in Virtuozzo format XXXXXXXXXXXX).

Syntax

```
PRL_RESULT PrlVmDevNet_GetMacAddress(
```

```
    PRL_HANDLE hVmDev,
    PRL_STR sMacAddress,
    PRL_UINT32_PTR pnMacAddressBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sMacAddress

[out] Pointer to a buffer that receives the MAC address (a UTF-8 encoded, null-terminated string).

pnMacAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_GetMacAddressCanonical (p. 396), PrlVmDevNet_SetMacAddress (p. 420)

PrlVmDevNet_GetNetAddresses Function

Obtains a list of IP addresses in the address/subnet_mask format which is assigned to a virtual network adapter on virtual machine startup.

Syntax

```
PRL_RESULT PrlVmDevNet_GetNetAddresses(
    PRL_HANDLE hVmDev,
    PRL_HANDLE_PTR phNetAddressesList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

phNetAddressesList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle containing the IP addresses.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate strings list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Returns the list of pairs "IP address/Subnet mask" which will be applied to the network adapter on the guest OS side (on guest start or immediately on config commit if guest already working and Virtuozzo tools with necessary functionality are installed). The result will be returned as list of strings (PHT_STRINGS_LIST (p. 357) type) of pairs "IP address/Subnet mask".

See Also

PrlVmDevNet_SetNetAddresses (p. 421)

PrlVmDevNet_GetSearchDomains Function

Obtains a lists of search domains which are assigned to a virtual network adapter on virtual machine startup.

Syntax

```
PRL_RESULT PrlVmDevNet_GetSearchDomains(
    PRL_HANDLE hVmDev,
    PRL_HANDLE_PTR phSearchDomainsList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

phSearchDomainsList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle containing the results.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate strings list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetSearchDomains (p. 423)

PrlVmDevNet_GetVirtualNetworkId Function

Obtains the virtual network ID assigned to a virtual network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_GetVirtualNetworkId(
```

```
    PRL_HANDLE hVmDev,
    PRL_STR sVirtualNetworkId,
    PRL_UINT32_PTR pnVirtualNetworkIdBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sVirtualNetworkId

[out] A pointer to a buffer that receives the virtual network ID (a UTF-8 encoded, null-terminated string).

pnVirtNetIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_SetVirtualNetworkId (p. 424)

PrlVmDevNet_IsAutoApply Function

Determines if the specified network adapter is configured to automatically apply network settings inside guest.

Syntax

```
PRL_RESULT PrlVmDevNet_IsAutoApply(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbAutoApply
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pbAutoApply

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the adapter is configured to automatically apply network settings. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_IsConfigureWithDhcp Function

Determines if the specified network adapter is configured through DHCP on the guest OS side.

Syntax

```
PRL_RESULT PrlVmDevNet_IsConfigureWithDhcp(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbConfigureWithDhcp
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pbConfigureWithDhcp

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the adapter is configured through DHCP. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_IsConfigureWithDhcpIPv6 Function

Determines if the specified network adapter is configured through DHCP (IPv6) on the guest OS side.

Syntax

```
PRL_RESULT PrlVmDevNet_IsConfigureWithDhcpIPv6(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbConfigureWithDhcp
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pbConfigureWithDhcp

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the adapter is configured through DHCP. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_IsFirewallEnabled Function

Gets state switched on or off of the adapter firewall.

Syntax

```
PRL_RESULT PrlVmDevNet_IsFirewallEnabled(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pbEnabled

[out] A pointer to a buffer that receives the adapter firewall state.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_IsPktFilterPreventIpSpoof Function

Determines if the specified network adapter is configured to filter net packages and send packages only from IP addresses in the adapter NetAdrresses list.

Syntax

```
PRL_RESULT PrlVmDevNet_IsPktFilterPreventIpSpoof(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbPktFilterPreventIpSpoof
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pbPktFilterPreventMacSpoof

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that adapter is allowed to send packages only from IP addresses in the adapter NetAdrresses list. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_IsPktFilterPreventMacSpoof Function

Determines if the specified network adapter is configured to filter net packages and send packages only from its own MAC address.

Syntax

```
PRL_RESULT PrlVmDevNet_IsPktFilterPreventMacSpoof(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbPktFilterPreventMacSpoof
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

pbPktFilterPreventMacSpoof

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that adapter is allowed to send packages only from its own MAC address. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_SetAdapterType Function

Allows to manually assign adapter type to a network adapter.

Syntax

```
PRL_RESULT PrlVmDevNet_SetAdapterType(
    PRL_HANDLE hVmDev,
    PRL_VM_NET_ADAPTER_TYPE nAdapterType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

nAdapterType

The new network adapter type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_GetAdapterType (p. 401)

PrlVmDevNet_SetAutoApply Function

Allows to specify whether a network adapter in a virtual machine should be automatically configured.

Syntax

```
PRL_RESULT PrlVmDevNet_SetAutoApply(
    PRL_HANDLE hVmDev,
    PRL_BOOL bAutoApply
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bAutoApply

Specify PRL_TRUE (p. 1652) to configure the adapter automatically. Specified PRL_FALSE (p. 1626) to configure the network settings manually in a virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in a target virtual machine.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

PrlVmDevNet_SetBoundAdapterIndex Function

Sets the index of the adapter to which the specified virtual adapter should be bound.

Syntax

```
PRL_RESULT PrlVmDevNet_SetBoundAdapterIndex(
    PRL_HANDLE hVmDev,
    PRL_INT32 nIndex
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

nIndex

The index of the adapter to which the specified virtual adapter should be bound. If you would like to bind the virtual adapter to the default adapter, specify -1 (negative one).

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid net VM device handle or handle of already removed net VM device was passed PRL_ERR_SUCCESS (p. 1365) - operation successfully completed

Remarks

In a bridged networking mode, a network adapter inside a virtual machine is bound to an adapter on the host. This function allows to set the index of the adapter from the host.

See Also

PrlVmDevNet_GetBoundAdapterIndex (p. 402)

PrlVmDevNet_SetBoundAdapterName Function

Sets the name of the network adapter to which a virtual network adapter inside a virtual machine will bind.

Syntax

```
PRL_RESULT PrlVmDevNet_SetBoundAdapterName(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewBoundAdapterName
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sNewBoundAdapterName

The name of the adapter to bind to. The adapter must exist on the host. To get the names of the available adapters from the host, use the PrlSrvCfg_GetNetAdapter (p. 332) function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

In a bridged networking mode, a network adapter inside a virtual machine is bound to an adapter on the host. This function allows to set the name of the adapter to bind to.

See Also

PrlVmDevNet_GetBoundAdapterName (p. 402)

PrlVmDevNet_SetConfigureWithDhcp Function

Allows to specify whether a network adapter in a virtual machine should be configured through DHCP or manually.

Syntax

```
PRL_RESULT PrlVmDevNet_SetConfigureWithDhcp(
    PRL_HANDLE hVmDev,
    PRL_BOOL bConfigureWithDhcp
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bConfigureWithDhcp

Specify PRL_TRUE (p. 1652) to configure the adapter through DHCP. Specified PRL_FALSE (p. 1626) to configure the network settings manually.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetConfigureWithDhcpIPv6 Function

Allows to specify whether a network adapter in a virtual machine should be configured through DHCP (IPv6) or manually.

Syntax

```
PRL_RESULT PrlVmDevNet_SetConfigureWithDhcpIPv6(
    PRL_HANDLE hVmDev,
    PRL_BOOL bConfigureWithDhcp
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bConfigureWithDhcp

Specify PRL_TRUE (p. 1652) to configure the adapter through DHCP. Specified PRL_FALSE (p. 1626) to configure the network settings manually.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetDefaultGateway Function

Allows to manually set the default gateway address for a network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetDefaultGateway(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewDefaultGateway
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sNewDefaultGateway

The new default gateway address. The specified string must be encoded using UTF-8 and must be terminated with the binary zero character.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_GetDefaultGateway (p. 403) PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetDefaultGatewayIPv6 Function

Allows to manually set the default gateway address (IPv6) for a network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetDefaultGatewayIPv6(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewDefaultGateway
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sNewDefaultGateway

The new default gateway address. The specified string must be encoded using UTF-8 and must be terminated with the binary zero character.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_GetDefaultGateway (p. 403) PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetDnsServers Function

Allows to manually assign DNS servers to a network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetDnsServers(
    PRL_HANDLE hVmDev,
    PRL_HANDLE hDnsServersList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

hDnsServersList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of DNS servers to be assigned to the network adapter. Can be PRL_INVALID_HANDLE (p. 1627) if it's necessary to specify an empty list.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network adapter specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetFirewallDefaultPolicy Function

Sets an adapter firewall default policy.

Syntax

```
PRL_RESULT PrlVmDevNet_SetFirewallDefaultPolicy(
    PRL_HANDLE hVmDev,
    PRL_FIREWALL_DIRECTION nDirection,
    PRL_FIREWALL_POLICY nPolicy
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

nDirection

Direction of a firewall - to/from the virtual network adapter

nPolicy

An adapter firewall default policy.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Policy values are defined in the PRL_FIREWALL_POLICY (p. 1060) enumeration.

PrlVmDevNet_SetFirewallEnabled Function

Sets state switched on or off of the adapter firewall.

Syntax

```
PRL_RESULT PrlVmDevNet_SetFirewallEnabled(
    PRL_HANDLE hVmDev,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bEnabled

The adapter firewall state.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_SetFirewallRuleList Function

Sets the firewall rule entries list.

Syntax

```
PRL_RESULT PrlVmDevNet_SetFirewallRuleList(
    PRL_HANDLE hVmDev,
    PRL_FIREWALL_DIRECTION nDirection,
    PRL_HANDLE hRuleList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

nDirection

Direction of a firewall - to/from the virtual network adapter

hRuleList

The list of PHT_FIREWALL_RULE (p. 95) handles.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a port forward entry object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_SetMacAddress Function

Allows to manually set MAC address to a network adapter inside a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetMacAddress(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewMacAddress
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sNewMacAddress

The new MAC address. The specified string must be encoded using UTF-8 and must be terminated with a binary zero character. The MAC address value cannot contain any delimiters. The following is an example of a valid MAC address: "001731EE6FBC".

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function if you would like to set a specific MAC address to an adapter. To generate an address and to set it to an adapter automatically, use the PrlVmDevNet_GenerateMacAddr (p. 401) function.

See Also

PrlVmDevNet_GetMacAddress (p. 406)

PrlVmDevNet_SetNetAddresses Function

Allows to manually set IP addresses/Subnet masks to a network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetNetAddresses(
    PRL_HANDLE hVmDev,
    PRL_HANDLE hNetAddressesList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

hNetAddressesList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of "IP address/Subnet mask" pairs to be applied to the network adapter. Can be PRL_INVALID_HANDLE (p. 1627) if it's necessary to specify an empty list.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network adapter specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetPktFilterPreventIpSpoof Function

Allows to specify whether a network adapter in a virtual machine should perform the filtration by IP addresses.

Syntax

```
PRL_RESULT PrlVmDevNet_SetPktFilterPreventIpSpoof(
    PRL_HANDLE hVmDev,
    PRL_BOOL bPktFilterPreventIpSpoof
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bPktFilterPreventMacSpoof

Specify PRL_TRUE (p. 1652) to send packages only from addresses in the adapter NetAdrresses list. Specify PRL_FALSE (p. 1626) to switch off filtration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_SetPktFilterPreventMacSpoof Function

Allows to specify whether a network adapter in a virtual machine should filter packets through MAC address spoofing.

Syntax

```
PRL_RESULT PrlVmDevNet_SetPktFilterPreventMacSpoof(
    PRL_HANDLE hVmDev,
    PRL_BOOL bPktFilterPreventMacSpoof
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bPktFilterPreventMacSpoof

Specify PRL_TRUE (p. 1652) to send packages only from its own MAC address. Specify PRL_FALSE (p. 1626) to turn off filtration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_SetPktFilterPreventPromisc Function

Allows to specify whether a network adapter in a virtual machine should receive packets sent not to VM.

Syntax

```
PRL_RESULT PrlVmDevNet_SetPktFilterPreventPromisc(
    PRL_HANDLE hVmDev,
    PRL_BOOL bPktFilterPreventPromisc
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

bPktFilterPreventPromisc

Specify PRL_TRUE (p. 1652) to prevent VM to receive packets that were sent not to VM. Specify PRL_FALSE (p. 1626) to switch off filtration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDevNet_SetSearchDomains Function

Allows to manually assign search domains to a network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetSearchDomains(
    PRL_HANDLE hVmDev,
    PRL_HANDLE hSearchDomainsList
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

hSearchDomainsList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of search domains to be assigned to the network adapter. Can be PRL_INVALID_HANDLE (p. 1627) iif it's necessary to specify an empty list.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network adapter specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmDevNet_SetAutoApply (p. 413)

PrlVmDevNet_SetVirtualNetworkId Function

Allows to set the virtual network ID for a network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmDevNet_SetVirtualNetworkId(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewVirtualNetworkId
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) identifying the virtual network adapter.

sNewVirtualNetworkId

The new virtual network ID. The specified string must be encoded using UTF-8 and must be terminated with the binary zero character. Can be null or empty in order to drop virtual network ID device field.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevNet_GetVirtualNetworkId (p. 408)

PHT_VIRTUAL_DEV_OPTICAL_DISK

The PHT_VIRTUAL_DEV_OPTICAL_DISK handle contains information about a virtual optical disks in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetOpticalDisk (p. 593) function call.

See Also

Common Virtual Device Functions (p. 752)

PHT_VIRTUAL_DEV_PARALLEL_PORT

The PHT_VIRTUAL_DEV_PARALLEL_PORT handle contains information about a virtual parallel port in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetParallelPort (p. 597) function call.

See Also

Common Virtual Device Functions (p. 752)

PHT_VIRTUAL_DEV_SERIAL_PORT

The PHT_VIRTUAL_DEV_SERIAL_PORT handle is used to manage serial ports in a virtual machine. The handle is obtained by making the PrlVmCfg_CreateVmDev (p. 551) or the PrlVmCfg_GetSerialPort (p. 600) function call.

Description

The following describes how to create new or modify an existing serial port on a virtual machine.

First, you have to set the emulation type using the PrlVmDev_SetEmulatedType (p. 769) function. The available types are:

- PDT_USE_REAL_DEVICE - real port.
- PDT_USE_OUTPUT_FILE - output file.
- PDT_USE_OTHER - socket.

Second, you must call the PrlVmDev_SetSysName (p. 775) and the PrlVmDev_SetFriendlyName (p. 770) functions. These functions are used differently depending on the emulation type that you chose in the first step.

Real port

1. Choose a real port and pass its ID to the PrlVmDev_SetSysName (p. 775) function.
2. Pass the physical port name to the PrlVmDev_SetFriendlyName (p. 770) function.

Output file

1. Choose an output file name and path, then pass it to both functions as a parameter.

Socket

1. Choose a socket name and path, then pass it to both functions as a parameter.
2. Call the PrlVmDevSerial_SetSocketMode (p. 427) function to set the socket mode.

Functions

	Name	Description
	PrlVmDevSerial_GetSocketMode (p. 426)	Returns the socket mode of a virtual serial port.
	PrlVmDevSerial_SetSocketMode (p. 427)	Sets the socket mode for a virtual serial port.

See Also

Common Virtual Device Functions (p. 752)

PrlVmDevSerial_GetSocketMode Function

Returns the socket mode of a virtual serial port.

Syntax

```
PRL_RESULT PrlVmDevSerial_GetSocketMode(
    PRL_HANDLE hVmDev,
    PRL_SERIAL_PORT_SOCKET_OPERATION_MODE_PTR pnSocketMode
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425) identifying the serial port.

pnSocketMode

[out] Pointer to a variable that receives the socket mode.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function to get the socket mode when the emulation type of the port is "socket". To get the port's emulation type, use the PrlVmDev_GetEmulatedType (p. 757) function.

See Also

PrlVmDevSerial_SetSocketMode (p. 427)

PrlVmDevSerial_SetSocketMode Function

Sets the socket mode for a virtual serial port.

Syntax

```
PRL_RESULT PrlVmDevSerial_SetSocketMode(
    PRL_HANDLE hVmDev,
    PRL_SERIAL_PORT_SOCKET_OPERATION_MODE nSocketMode
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425) identifying the serial port.

nSocketMode

The socket mode emulation type to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function to set the socket mode when the emulation type of the port is "socket". To get the port's emulation type, use the PrlVmDev_GetEmulatedType (p. 757) function.

See Also

PrlVmDevSerial_GetSocketMode (p. 426)

PHT_VIRTUAL_DEV_SOUND

The PHT_VIRTUAL_DEV_SOUND handle is used to manage sound devices in a virtual machine.

Functions

	Name	Description
	PrlVmDevSound_GetMixerDev (p. 428)	Returns mixer device string for the specified sound device.
	PrlVmDevSound_GetOutputDev (p. 428)	Returns output device string for the specified sound device.
	PrlVmDevSound_SetMixerDev (p. 429)	Sets mixer device string for the specified sound device.
	PrlVmDevSound_SetOutputDev (p. 430)	Sets output device string for the specified sound device.

See Also

Common Virtual Device Functions (p. 752)

PrlVmDevSound_GetMixerDev Function

Returns mixer device string for the specified sound device.

Syntax

```
PRL_RESULT PrlVmDevSound_GetMixerDev(
    PRL_HANDLE hVmDev,
    PRL_STR sMixerDev,
    PRL_UINT32_PTR pnMixerDevBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_SOUND (p. 427) identifying the sound device.

sMixerDev

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnMixerDevBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevSound_SetMixerDev (p. 429)

PrlVmDevSound_GetOutputDev Function

Returns output device string for the specified sound device.

Syntax

```
PRL_RESULT PrlVmDevSound_GetOutputDev(
    PRL_HANDLE hVmDev,
    PRL_STR sOutputDev,
    PRL_UINT32_PTR pnOutputDevBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_SOUND (p. 427) identifying the sound device.

sOutputDev

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnOutputDevBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevSound_SetOutputDev (p. 430)

PrlVmDevSound_SetMixerDev Function

Sets mixer device string for the specified sound device.

Syntax

```
PRL_RESULT PrlVmDevSound_SetMixerDev(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewMixerDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_SOUND (p. 427) identifying the sound device.

sNewMixerDev

Mixer device string to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevSound_GetMixerDev (p. 428)

PrlVmDevSound_SetOutputDev Function

Sets output device string for the specified sound device.

Syntax

```
PRL_RESULT PrlVmDevSound_SetOutputDev(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewOutputDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_SOUND (p. 427) identifying the sound device.

sNewOutputDev

Output device string to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevSound_GetOutputDev (p. 428)

PHT_VIRTUAL_DEV_USB_DEVICE

The PHT_VIRTUAL_DEV_USB handle is used to managing USB devices in a virtual machine.

Functions

	Name	Description
	PrlVmDevUsb_GetAutoconnectOption (p. 431)	Obtains USB controller autoconnect device option.
	PrlVmDevUsb_SetAutoconnectOption (p. 431)	Sets USB controller autoconnect device option.

See Also

Common Virtual Device Functions (p. 752)

PrlVmDevUsb_GetAutoconnectOption Function

Obtains USB controller autoconnect device option.

Syntax

```
PRL_RESULT PrlVmDevUsb_GetAutoconnectOption(
    PRL_HANDLE hVmDev,
    PRL_USB_DEVICE_AUTO_CONNECT_OPTION_PTR pnAutoconnectOption
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_USB_DEVICE (p. 430) identifying the USB device.

pnAutoconnectOption

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevUsb_SetAutoconnectOption (p. 431)

PrlVmDevUsb_SetAutoconnectOption Function

Sets USB controller autoconnect device option.

Syntax

```
PRL_RESULT PrlVmDevUsb_SetAutoconnectOption(
    PRL_HANDLE hVmDev,
    PRL_USB_DEVICE_AUTO_CONNECT_OPTION nAutoconnectOption
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle of type PHT_VIRTUAL_DEV_USB_DEVICE (p. 430) identifying the USB device.

nAutoconnectOption

Autoconnect option to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevUsb_GetAutoconnectOption (p. 431)

PHT_VIRTUAL_DISK

The PHT_VIRTUAL_DISK handle is used to create, configure, and manage Virtuozzo virtual disks. A virtual disk is comprised of a set of files including a configuration file containing information about the disk parameters (number of heads, cylinders, sectors, etc.) and the files that are used for data storage. Once created, a virtual disk can be connected to a virtual machine and recognized by it, just like a physical hard disk can be installed in a physical box. To create a new virtual disk use PrlDisk_CreateDisk or PrlDisk_CreateDisk_Local (p. 433). To obtain a handle to an existing virtual disk, use PrlDisk_OpenDisk or PrlDisk_OpenDisk_Local (p. 435). You must always free the handle using the PrlDisk_Free (p. 434) function when the handle is no longer needed. Failure to do so will result in memory leak.

Notes

The functionality provided by this handle offers more low-level control over new and existing virtual disks. You can use it to create standalone new disks and then decide on the virtual machine to which the disk should be connected. You can also manage existing disks, including those used by virtual machines. If, however, all you want to do is create a virtual machine and a virtual disk at the same time, or add a virtual disk to a virtual machine in a few easy steps, you can use the functionality provided by the PHT_VIRTUAL_MACHINE (p. 442) and PHT_VM_CONFIGURATION (p. 537) handles. You can also consult **Virtualization SDK Programmer's Guide** for a detailed explanation and code samples on how to create a new virtual machine and how to add a virtual disk to an existing virtual machine.

Functions

	Name	Description
	PrlDisk_CreateDisk_Local (p. 433)	This function can be used to create a new virtual disk locally without being connected to the Virtuozzo Service.
	PrlDisk_Free (p. 434)	Frees the PHT_VIRTUAL_DISK handle.
	PrlDisk_GetDiskInfo (p. 434)	Obtains the information about the specified virtual disk.
	PrlDisk_OpenDisk_Local (p. 435)	Obtains a handle of type PHT_VIRTUAL_DISK identifying an existing virtual disk on a local machine.
	PrlDisk_Read (p. 436)	Reads data from the specified virtual disk.
	PrlDisk_SwitchToState (p. 436)	Reverts a virtual disk to the specified snapshot.
	PrlDisk_WaitForCompletion (p. 437)	Waits for the background thread to finish executing before returning control to the client.
	PrlDisk_Write (p. 438)	Writes data to the specified virtual disk.
	PrlDiskMap_GetGranularity (p. 438)	Reports the size (in bytes) of a block mapped by a single bit in the map.
	PrlDiskMap_GetSize (p. 439)	Reports the number of significant bits in the map.
	PrlDiskMap_Read (p. 439)	Returns bits from the blocks map.
	PrlDiskOpenPolicy_CreateOffset (p. 440)	Obtains a handle of type PHT_VIRTUAL_DISK_OPEN_POLICY identifying an offset policy.

PrlDisk_CreateDisk_Local Function

This function can be used to create a new virtual disk locally without being connected to the Virtuozzo Service.

Syntax

```
PRL_RESULT PrlDisk_CreateDisk_Local(
    PRL_HANDLE_PTR pHandle,
    PRL_CONST_STR pDiskName,
    PRL_CONST_DISK_PARAMETERS_PTR pParams,
    const PRL_DISK_CALLBACK pCallback,
    PRL_CONST_VOID_PTR pAdditional
);
```

File

PrlDisk.h

Parameters

pHandle

[out] A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the new virtual disk. You must always free the handle using the PrlDisk_Free (p. 434) function when the handle is no longer needed. Failure to do so will result in memory leak.

pDiskName

Disk name in UTF-8 format.

pParams

Disk configuration parameters.

pCallback

A pointer to a callback function or NULL pointer. This parameter allows to call the function asynchronously. If this parameter contains a pointer to a user-implemented callback function, the PrlDisk_CreateDisk function will return immediately and the operation will continue in the background thread. The background thread will be calling the callback function passing the operation progress information to it. See PRL_DISK_CALLBACK (p. 779) for more information. Using a callback might be helpful when the disk creation is expected to take a long time. For example, you may want to use a callback when you are creating a large disk and allocating all space for it right away.

If this parameter contains a NULL pointer, the function will be called synchronously.

pAdditional

A pointer to a buffer containing user-defined data to pass to the callback function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The function creates a new virtual disk and obtains a handle of type PHT_VIRTUAL_DISK (p. 432) identifying the new disk. The handle can then be used to perform other operations on the disk if necessary (see other functions from this group). If you would like to create a new virtual disk on a remote host, use the PrlDisk_CreateDisk function.

See Also

PrlDisk_CreateDisk

PrlDisk_Free (p. 434)

PrlDisk_Free Function

Frees the PHT_VIRTUAL_DISK (p. 432) handle.

Syntax

```
PRL_RESULT PrlDisk_Free(
    const PRL_HANDLE Handle
);
```

File

PrlDisk.h

Parameters

Handle

A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

This function must be called to free the PHT_VIRTUAL_DISK (p. 432) handle when it is no loner needed. Failure to do so will result in a memory leak.

PrlDisk_GetDiskInfo Function

Obtains the information about the specified virtual disk.

Syntax

```
PRL_RESULT PrlDisk_GetDiskInfo(
    const PRL_HANDLE Handle,
    PRL_DISK_PARAMETERS_PTR pDiskParameters,
    PRL_UINT32_PTR BufferSize
);
```

File

PrlDisk.h

Parameters

Handle

A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk.

pDiskParameters

[out] A pointer to a buffer that receives the disk information.

BufferSize

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlDisk_OpenDisk_Local Function

Obtains a handle of type PHT_VIRTUAL_DISK (p. 432) identifying an existing virtual disk on a local machine.

Syntax

```
PRL_RESULT PrlDisk_OpenDisk_Local(
    PRL_HANDLE_PTR pHandle,
    PRL_CONST_STR pDiskName,
    const PRL_DISK_OPEN_FLAGS OpenFlags,
    PRL_HANDLE hPolicyList
);
```

File

PrlDisk.h

Parameters

pHandle

[out] A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk. You must always free the handle using the PrlDisk_Free (p. 434) function when the handle is no longer needed. Failure to do so will result in memory leak.

pDiskName

The disk name for which to obtain the handle, in UTF-8 format. Disk name is the same as the bundle name (Mac) or the directory name (Windows, Linux) containing the virtual disk files.

OpenFlags

Disk open parameters. See PRL_DISK Macros (p. 1128) for available options.

hPolicyList

A handle of type PHT_HANDLE_LIST containing handles of type PHT_VIRTUAL_DISK_OPEN_POLICY representing disk open policies. May be NULL.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when you want to view or modify the configuration of an existing virtual disk. Once you obtain the disk handle, use its functions to perform the desired disk manipulations. This function can be used locally without a Virtuozzo Service connection.

See Also

PrlDisk_OpenDisk

PrlDisk_Free (p. 434)

PrlDisk_Read Function

Reads data from the specified virtual disk.

Syntax

```
PRL_RESULT PrlDisk_Read(
    const PRL_HANDLE Handle,
    PRL_VOID_PTR pBlock,
    const PRL_UINT32 uiSize,
    const PRL_UINT64 uiBlockOffset
);
```

File

PrlDisk.h

Parameters

Handle

A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk.

pBlock

A pointer to sector/page-aligned data to read (depends on the host operating system).

uiSize

The size of the data to read.

uiBlockOffset

Offset of the block on the disk, in sectors.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlDisk_Write (p. 438)

PrlDisk_SwitchToState Function

Reverts a virtual disk to the specified snapshot.

Syntax

```
PRL_RESULT PrlDisk_SwitchToState(
    const PRL_HANDLE Handle,
    PRL_CONST_STR pDirectoryName,
    PRL_CONST_GUID_PTR pUid,
    const PRL_STATES_CALLBACK pCallback,
    PRL_VOID_PTR pParameter
);
```

File

PrlDisk.h

Parameters

Handle

A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk.

pDirectoryName

Directory name. This parameter is not currently used.

pUid

The universally unique ID (UUID) assigned to the disk. Use the PrlDisk_GetSnapshotsTree function to obtain the snapshot tree with UUIDs of individual snapshots.

pCallback

A pointer to a callback function or NULL pointer. If the parameter contains a valid pointer to the callback, the function will be called asynchronously. If it contains a NULL pointer, the function will be called synchronously.

pParameter

A pointer to a buffer containing user-defined data to pass to the callback function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlDisk_CreateState

PrlDisk_DeleteState

PrlDisk_GetSnapshotsTree

PrlDisk_WaitForCompletion Function

Waits for the background thread to finish executing before returning control to the client.

Syntax

```
PRL_RESULT PrlDisk_WaitForCompletion(
    const PRL_HANDLE Handle
);
```

File

PrlDisk.h

Parameters

Handle

A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function with any of the PHT_VIRTUAL_DISK (p. 432) disk functions that can be executed asynchronously. When a function is called asynchronously, it returns immediately. The actual task that was initiated by the call is performed in the background thread. By calling the PrlDisk_WaitForCompletion function, you will suspend the main thread execution until the background thread is finished processing the task.

PrlDisk_Write Function

Writes data to the specified virtual disk.

Syntax

```
PRL_RESULT PrlDisk_Write(
    const PRL_HANDLE Handle,
    PRL_CONST_VOID_PTR pBlock,
    const PRL_UINT32 uiSize,
    const PRL_UINT64 uiBlockOffset
);
```

File

PrlDisk.h

Parameters

Handle

A handle of type PHT_VIRTUAL_DISK (p. 432) identifying the virtual disk.

pBlock

A pointer to the sector/page-aligned data to write (depends on the host operating system).

uiSize

Size of the data to write.

uiBlockOffset

Offset of the block on the disk, in sectors.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid argument values.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlDisk_Read (p. 436)

PrlDiskMap_GetGranularity Function

Reports the size (in bytes) of a block mapped by a single bit in the map.

Syntax

```
PRL_RESULT PrlDiskMap_GetGranularity(
    PRL_HANDLE hMap,
    PRL_UINT32_PTR pnSize
);
```

File

PrlDisk.h

Parameters

hMap

A handle of type PHT_VIRTUAL_DISK_MAP identifying the changes map.

phSize

A pointer to a variable which receives the result.

Returns

PRL_RESULT.

PrlDiskMap_GetSize Function

Reports the number of significant bits in the map.

Syntax

```
PRL_RESULT PrlDiskMap_GetSize(
    PRL_HANDLE hMap,
    PRL_UINT32_PTR pnSize
);
```

File

PrlDisk.h

Parameters

hMap

A handle of type PHT_VIRTUAL_DISK_MAP identifying the changes map.

phSize

A pointer to a variable which receives the result.

Returns

PRL_RESULT.

PrlDiskMap_Read Function

Returns bits from the blocks map.

Syntax

```
PRL_RESULT PrlDiskMap_Read(
    PRL_HANDLE hMap,
    PRL_VOID_PTR pBuffer,
```

```
    PRL_UINT32_PTR pnCapacity
);
```

File

PrlDisk.h

Parameters

hMap

A handle of type PHT_VIRTUAL_DISK_MAP identifying the changes map.

pBuffer

A pointer to a store.

pnCapacity

A pointer to a variable holding the size of the buffer and receiving the number of bytes actually written.

Returns

PRL_RESULT.

PrlDiskOpenPolicy_CreateOffset Function

Obtains a handle of type PHT_VIRTUAL_DISK_OPEN_POLICY identifying an offset policy.

Syntax

```
PRL_RESULT PrlDiskOpenPolicy_CreateOffset(
    PRL_HANDLE_PTR pHandle,
    PRL_UINT64 uiOffset
);
```

File

PrlDisk.h

Parameters

pHandle

[out] A handle of type PHT_VIRTUAL_DISK_OPEN_POLICY identifying the offset policy. You must always free the handle using the PrlHandle_Free (p. 748) function when the handle is no longer needed. Failure to do so will result in memory leak.

uiOffset

Offset value in bytes. This value is added to each read/write operation offset.

Remarks

This policy allows to automatically increasing the offsets in PrlDisk_Read (p. 436)/Write by specified value.

PHT_VIRTUAL_DISK_ENCRYPTION

Functions

	Name	Description
	PrlVmDevHdEncryption_GetKeyId (p. 441)	Return disk encryption key identifier.

	PrlVmDevHdEncryption_SetKeyId (p. 441)	Set disk encryption key identifier.

PrlVmDevHdEncryption_GetKeyId Function

Return disk encryption key identifier.

Syntax

```
PRL_RESULT PrlVmDevHdEncryption_GetKeyId(
    PRL_HANDLE hEncryption,
    PRL_STR sKeyId,
    PRL_UINT32_PTR pnKeyIdBufLength
);
```

File

PrlApiVm.h

Parameters

hEncryption

A valid handle of type PHT_VIRTUAL_DISK_ENCRYPTION (p. 440) identifying the virtual hard disk encryption parameters. sKeyID [out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnKeyIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHdEncryption_SetKeyId (p. 441) PrlVmDevHd_GetEncryption (p. 384)

PrlVmDevHdEncryption_SetKeyId Function

Set disk encryption key identifier.

Syntax

```
PRL_RESULT PrlVmDevHdEncryption_SetKeyId(
    PRL_HANDLE hEncryption,
    PRL_CONST_STR sKeyId
);
```

File

PrlApiVm.h

Parameters

hEncryption

A valid handle of type PHT_VIRTUAL_DISK_ENCRYPTION (p. 440) identifying the virtual hard disk.

sKeyId

[in] Key identifier - an UTF-8 encoded string with null '0' symbol at the end.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDevHdEncryption_GetKeyId (p. 441) PrlVmDevHd_GetEncryption (p. 384)

PHT_VIRTUAL_MACHINE

PHT_VIRTUAL_MACHINE is the main virtual machine management handle. A handle to an existing virtual machine is obtained from the PHT_JOB (p. 138) object returned by the PrlSrv_GetVmList (p. 264) function. To create a new handle, call the PrlSrv_CreateVm (p. 232) function.

Functions

	Name	Description
	PrlCt_Reinstall (p. 445)	Reinstall Container Recreates Container's root virtual disk, installs all applications previously installed from application templates, copies credentials from the old container (unless the REINSTALL_RESET_PWDB flag is specified), and renames the old root directory '/' to '/old' (unless the REINSTALL_SKIP_BACKUP flag is specified).
	PrlVm_AuthWithGuestSecurityDb (p. 446)	Performs an attempt to authenticate the user through the guest OS security database.
	PrlVm_BeginEdit (p. 446)	The PrlVm_BeginEdit function is used in modifying a virtual machine configuration.
	PrlVm_CancelCompact (p. 448)	Finishes process of optimization of virtual hard disk.
	PrlVm_CancelConvertDisks (p. 448)	Finishes process of conversion of the virtual hard disk(s).
	PrlVm_CaptureScreen (p. 449)	%VM_ONLY% Captures a screen area of a remote virtual machine desktop in PNG format, scales is down to the specified size, and puts it into a buffer which is then returned to the client.
	PrlVm_ChangeSid (p. 450)	Generates and assigns a new Windows SID to the specified virtual machine running Windows guest OS.
	PrlVm_Clone (p. 450)	Clones an existing virtual machine.
	PrlVm_CloneEx (p. 451)	Clones an existing virtual machine.
	PrlVm_CloneWithUuid (p. 452)	Clones an existing virtual machine.
	PrlVm_Commit (p. 453)	The PrlVm_Commit function is used in modifying a virtual machine configuration.

	PrlVm_CommitEncryption (p. 454)	The PrlVm_CommitEncryption function is used in modifying a virtual machine disk encryption configuration.
	PrlVm_CommitEx (p. 455)	The PrlVm_CommitEx function is the same as PrlVm_Commit (p. 453) function but with the flags
	PrlVm_Compact (p. 456)	Start process of optimization of virtual hard disk.
	PrlVm_Connect (p. 456)	Connect to remote desktop.
	PrlVm_ConvertDisks (p. 457)	Start process conversion of the virtual hard disk(s).
	PrlVm_CreateEvent (p. 458)	Creates VM event that bint to current VM.
	PrlVm_CreateSnapshot (p. 458)	Creates a virtual machine snapshot.
	PrlVm_CreateSnapshotEx (p. 459)	Creates a virtual machine snapshot.
	PrlVm_CreateUnattendedFloppy (p. 460)	Creates a floppy disk image in the virtual machine home directory for unattended Windows installation.
	PrlVm_Delete (p. 461)	Deletes the specified virtual machine from the host.
	PrlVm_DeleteSnapshot (p. 461)	Deleted the specified virtual machine snapshot.
	PrlVm_Disconnect (p. 462)	Disconnect from remote desktop.
	PrlVm_DropSuspendedState (p. 463)	Resets a suspended virtual machine.
	PrlVm_GenerateVmDevFilename (p. 463)	Automatically generates a name for a virtual device which will be unique within a home directory of the specified virtual machine.
	PrlVm_GetConfig (p. 464)	Obtains a handle of type PHT_VM_CONFIGURATION (p. 537).
	PrlVm_GetPackedProblemReport (p. 465)	The PrlVm_GetPackedProblemReport function obtains a Packed problem report on abnormal virtual machine termination.
	PrlVm_GetPerfStats (p. 465)	Obtains a handle of type PHT_EVENT (p. 72) containing the virtual machine performance statistics.
	PrlVm_GetProblemReport (p. 466)	The PrlVm_GetProblemReport function obtains a problem report on abnormal virtual machine termination.
	PrlVm_GetQuestions (p. 467)	The PrlVm_GetQuestions function allows to synchronously receive questions from a Virtuozzo Service.
	PrlVm_GetSnapshotsTree (p. 467)	Retrieves snapshot information for the specified virtual machine.
	PrlVm_GetSnapshotsTreeEx (p. 468)	
	PrlVm_GetState (p. 468)	Obtains a handle of type PHT_VM_INFO (p. 719) containing the specified virtual machine state and access rights information.
	PrlVm_GetStatistics (p. 469)	Obtains a handle of type PHT_SYSTEM_STATISTICS (p. 360) containing the virtual machine resource usage information, including CPU(s), memory, disk drives, processes, user session, system uptime, network packets, etc.
	PrlVm_GetStatisticsEx (p. 470)	The PrlVm_GetStatisticsEx function is the same as PrlVm_GetStatistics (p. 469) function but with the flags
	PrlVm_GetSuspendedScreen (p. 470)	Retrieves the virtual machine screen state before it was suspending.
	PrlVm_GetToolsState (p. 471)	Obtains a handle of type PHT_VM_TOOLS_INFO (p. 724) containing information about the state of the Virtuozzo Tools package in a virtual machine.
	PrlVm_InitiateDevStateNotifications (p. 472)	This method lets to initiate sending notifications about devices states by VM.
	PrlVm_InstallTools (p. 472)	Begin the Virtuozzo Tools installation process.
	PrlVm_Lock (p. 473)	Lets to exclusively lock VM for current session.
	PrlVm_LoginInGuest (p. 473)	Creates a new console session or binds to an existing GUI session in a virtual machine.

	PrlVm_Migrate (p. 475)	Migrates an existing virtual machine to another host.
	PrlVm_MigrateCancel (p. 476)	Cancels the migration operation that was started with PrlVm_Migrate (p. 475) or PrlVm_MigrateEx (p. 476).
	PrlVm_MigrateEx (p. 476)	Migrates an existing virtual machine to another host.
	PrlVm_MigrateWithRename (p. 477)	Migrates an existing virtual machine to another host.
	PrlVm_MigrateWithRenameEx (p. 478)	Migrates an existing virtual machine to another host.
	PrlVm_Mount (p. 479)	Mount the specified virtual machine Virtual Disks.
	PrlVm_Move (p. 480)	Move bundle of virtual machine or private area of container.
	PrlVm_Pause (p. 481)	Pauses the specified virtual machine.
	PrlVm_RefreshConfig (p. 481)	The PrlVm_RefreshConfig function updates the configuration information contained in the specified virtual machine object (handle).
	PrlVm_RefreshConfigEx (p. 482)	The PrlVm_RefreshConfigEx function updates the configuration information contained in the specified virtual machine object (handle).
	PrlVm_Reg (p. 483)	See also PrlVm_RegEx (p. 485)
	PrlVm_RegEventHandler (p. 485)	Registers an event handler (callback function) with the virtual machine.
	PrlVm_RegEx (p. 485)	Creates a new virtual machine and registers it with the Virtuozzo Service.
	PrlVm_Reset (p. 486)	Resets the specified virtual machine.
	PrlVm_ResetUptime (p. 487)	Resets the virtual machine uptime counter (start date and counter itself).
	PrlVm_Restart (p. 488)	Restarts the operating system of the specified virtual machine.
	PrlVm_Restore (p. 488)	Restores the registered virtual machine.
	PrlVm_Resume (p. 489)	Resumes a virtual machine that was previously suspended with the PrlVm_Suspend (p. 496) call.
	PrlVm_SetConfig (p. 489)	The PrlVm_SetConfig is reserved and does nothing.
	PrlVm_SetUserPasswd (p. 490)	Allows to change the password of a guest operating system user.
	PrlVm_Start (p. 491)	Start the specified virtual machine.
	PrlVm_StartEx (p. 491)	Starts the specified virtual machine using the specified mode (see the PRL_VM_START_MODE (p. 950) enumeration).
	PrlVm_StartVncServer (p. 492)	Starts a VNC server for the specified virtual machine.
	PrlVm_Stop (p. 493)	Stop the specified virtual machine.
	PrlVm_StopEx (p. 493)	Stop the specified virtual machine.
	PrlVm_StopVncServer (p. 494)	Stops the VNC server that was previously started for a virtual machine.
	PrlVm_SubscribeToGuestStatistics (p. 495)	Subscribes to receive the specified virtual machine performance statistics on a periodic basis.
	PrlVm_SubscribeToPerfStats (p. 495)	Allows to subscribe to periodicaly receive virtual machine perfomance statistics.
	PrlVm_Suspend (p. 496)	Suspends the specified virtual machine.
	PrlVm_SwitchToSnapshot (p. 497)	Reverts the specified virtual machine to the specified snapshot.
	PrlVm_SwitchToSnapshotEx (p. 498)	Reverts the specified virtual machine to the specified snapshot.
	PrlVm_Umount (p. 498)	Unmount the specified virtual machine Virtual Disks.
	PrlVm_Unlock (p. 499)	Lets to unlock previously locked with PrlVm_Lock (p. 473) call VM.
	PrlVm_Unreg (p. 500)	Unregisters the specified virtual machine from the Virtuozzo Service.

	PrlVm_UnregEventHandler (p. 500)	Unregisters an event handler (callback function) that was previously registered using the PrlVm_RegEventHandler (p. 485) function.
	PrlVm_UnsubscribeFromGuestStatistics (p. 501)	Cancels the performance statistics subscription that was previously initiated using the PrlVm_SubscribeToGuestStatistics (p. 495) function.
	PrlVm_UnsubscribeFromPerfStats (p. 501)	Cancels the Virtuozzo Service performance statistics subscription that was started using the PrlVm_SubscribeToPerfStats (p. 495) function.
	PrlVm_UpdateSecurity (p. 502)	Updates access level for the specified virtual machine.
	PrlVm_UpdateSnapshotData (p. 503)	Modifies the specified virtual machine snapshot name and description.
	PrlVm_ValidateConfig (p. 504)	Validates the specified section of a virtual machine configuration.
	PrlVmCfg_IsHighResolutionEnabled (p. 505)	Determines disable or enable high resolution video mode for the specified virtual machine.
	PrlVmCfg_IsVerticalSynchronizationEnabled (p. 505)	Determines disable or enable vertical synchronization video mode for the specified virtual machine.
	PrlVmCfg_SetHighResolutionEnabled (p. 506)	Disables or enables high resolution video mode for the specified virtual machine.
	PrlVmCfg_SetVerticalSynchronizationEnabled (p. 506)	Disables or enables vertical synchronization video mode for the specified virtual machine.

PrlCt_Reinstall Function

Reinstall Container Recreates Container's root virtual disk, installs all applications previously installed from application templates, copies credentials from the old container (unless the REINSTALL_RESET_PWDB flag is specified), and renames the old root directory '/' to '/old' (unless the REINSTALL_SKIP_BACKUP flag is specified).

Syntax

```
PRL_HANDLE PrlCt_Reinstall(
    PRL_HANDLE hVm,
    PRL_CONST_STR sOsTemplate,
    PRL_UINT32 nFlags
);
```

File

PrlApiCt.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sOsTemplate

New OS template name.

nFlags

Bitset of flags - from enum PRL_CT_REINSTALL_OPTIONS (p. 1047)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are: PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_AuthWithGuestSecurityDb Function

Performs an attempt to authenticate the user through the guest OS security database.

Syntax

```
PRL_HANDLE PrlVm_AuthWithGuestSecurityDb(
    PRL_HANDLE hVm,
    PRL_CONST_STR sUserName,
    PRL_CONST_STR sUserPassword,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sUserName

User login name.

sUserPassword

User password (can be an empty string or NULL if the user doesn't have a password).

nFlags

A bitset of flags (should be 0 in the current implementation).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer (empty string) for user name was passed.

PRL_ERR_AUTHENTICATION_FAILED (p. 1190) - user authentication failed.

PRL_ERR_SUCCESS (p. 1365) - user authenticated successfully.

PrlVm_BeginEdit Function

The PrlVm_BeginEdit function is used in modifying a virtual machine configuration.

Syntax

```
PRL_HANDLE PrlVm_BeginEdit(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The PrlVm_BeginEdit and the PrlVm_Commit (p. 453) functions are used to detect collisions with other clients trying to modify the same virtual machine at the same time. The PrlVm_BeginEdit call timestamps the beginning of the editing operation. It does not lock the machine, so other clients can modify the same machine at the same time. When you are done making the changes, you must call the PrlVm_Commit (p. 453) function to save them in the virtual machine configuration. The function will verify that the configuration has not been modified by another client. If a collision is detected, your changes will be rejected. To ensure that you always work with the most recent configuration data, the PrlVm_BeginEdit function reloads it when called. If you first modified the data in your program and then called the PrlVm_BeginEdit function, your changes will be lost.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Example

The following example illustrates how to use PrlVm_BeginEdit and PrlVm_Commit (p. 453) functions.

```
PRL_HANDLE hJobBeginEdit;
PRL_HANDLE hJobCommit;
PRL_RESULT nJobRetCode;

// Timestamp the beginning of the configuration changes operation.
hJobBeginEdit = PrlVm_BeginEdit(hVm);

// Wait for the job to complete.
ret = PrlJob_Wait(hJobBeginEdit, 10000);

// Check for errors.
PrlJob_GetRetCode(hJobBeginEdit, &nJobRetCode);

if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJobBeginEdit);
    return nJobRetCode;
}

// The code modifying configuration parameters goes here...

// Commit the changes when done.
hJobCommit = PrlVm_Commit(hVm);

// Wait for the job to complete.
```

```
ret = PrlJob_Wait(hJobCommit, 10000);

// Check the results of the commit operation.
PrlJob_GetRetCode(hJobCommit, &nJobRetCode);

if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Commit error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJobCommit);
    return nJobRetCode;
}
```

PrlVm_CancelCompact Function

Finishes process of optimization of virtual hard disk.

Syntax

```
PRL_HANDLE PrlVm_CancelCompact(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

valid PRL_HANDLE (p. 820) describing the virtual machine

Returns

PRL_HANDLE (p. 820) to job that describing async operation result (PRL_INVALID_HANDLE (p. 1627) means not enough memory to instantiate async job represents object handle). Operation return code can be extracted with @ref PrlJob_GetRetCode (p. 143)() call. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle was passed PRL_ERR_UNINITIALIZED (p. 1373) - unexpected error: uninitialized with server connection or non identified VM object

Remarks

The machine should be registered on the server to do that.

PrlVm_CancelConvertDisks Function

Finishes process of conversion of the virtual hard disk(s).

Syntax

```
PRL_HANDLE PrlVm_CancelConvertDisks(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

valid PRL_HANDLE (p. 820) describing the virtual machine

nFlags

Reserved.

Returns

PRL_HANDLE (p. 820) to job that describing async operation result (PRL_INVALID_HANDLE (p. 1627) means not enough memory to instantiate async job represents object handle). Operation return code can be extracted with @ref PrlJob_GetRetCode (p. 143)() call. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle was passed PRL_ERR_UNINITIALIZED (p. 1373) - unexpected error: uninitialized with server connection or non identified VM object

Remarks

The machine should be registered on the server to do that.

PrlVm_CaptureScreen Function

%VM_ONLY%

Captures a screen area of a remote virtual machine desktop in PNG format, scales is down to the specified size, and puts it into a buffer which is then returned to the client.

Syntax

```
PRL_HANDLE PrlVm_CaptureScreen(
    PRL_HANDLE hVm,
    PRL_UINT32 nWidth,
    PRL_UINT32 nHeight,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nWidth

The width of the scaled down image, in pixels.

nHeight

The height of the scaled down image, in pixels.

nFlags

Reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

For local connections, the capture works almost instantly. For remote connections, the data must be transferred over network to the client machine, so it may take some time.

PrlVm_ChangeSid Function

Generates and assigns a new Windows SID to the specified virtual machine running Windows guest OS.

Syntax

```
PRL_HANDLE PrlVm_ChangeSid(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nFlags

Reserved for future use.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object. To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are: PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully. PRL_ERR_UNINITIALIZED (p. 1373) ??“ unknown PHT_VIRTUAL_MACHINE (p. 442) handle or invalid binding to server connection.

Remarks

This function is useful when cloning or migrating a virtual machine to another host to avoid conflicts. Please note that this functionality is applicable to Windows guests only. Virtuozzo Tools must be installed in the virtual machine. The machine must be stopped before calling this function.

PrlVm_Clone Function

Clones an existing virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Clone(
    PRL_HANDLE hVm,
    PRL_CONST_STR new_vm_name,
    PRL_CONST_STR new_vm_root_path,
    PRL_BOOL bCreateTemplate
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

new_vm_name

The name to use for the new virtual machine.

new_vm_root_path

Name and path of the directory where the new virtual machine should be created. To create the machine in a default directory, pass an empty string.

bCreateTemplate

Specifies whether the clone should be created as a virtual machine or as a template. To create a template, set this parameter value to PRL_TRUE (p. 1652). To create a virtual machine, set it to PRL_FALSE (p. 1626). When creating a virtual machine from a template, set the parameter value to PRL_FALSE (p. 1626).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function creates an exact copy of the specified virtual machine on the same host using the specified new name and location. In order to perform a clone operation, the original machine must be registered with the Virtuozzo Service. A clone can be created as a virtual machine or as a template. To create a template, set the bCreateTemplate parameter value to PRL_TRUE (p. 1652). The function can also be used to create new virtual machines from templates. In order to that, the hVm parameter must contain a handle to a template, the name and the path parameters must be set to desired values, and the bCreateTemplate parameter must be set to PRL_FALSE (p. 1626).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_CloneEx Function

Clones an existing virtual machine.

Syntax

```
PRL_HANDLE PrlVm_CloneEx(
    PRL_HANDLE hVm,
    PRL_CONST_STR new_vm_name,
    PRL_CONST_STR new_vm_root_path,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

new_vm_name

The name to use for the new virtual machine.

new_vm_root_path

Name and path of the directory where the new virtual machine should be created. To create the machine in a default directory, pass an empty string.

nFlags

Clone flags (PCVF_CLONE_TO_TEMPLATE, PCVF_CHANGE_SID, PCVF_LINKED_CLONE, PCVF_IMPORT_BOOT_CAMP). Also PACF_NON_INTERACTIVE_MODE can be used to specify non interactive session working mode (remote side won't ask questions during operation progress in this case). Please note that clone operation can take it's own exclusive lock of virtual machine (in change SID or linked clone modes) - so VM shouldn't be exclusively locked with PrlVm_Lock (p. 473) call before clone operation performed.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function creates an exact copy of the specified virtual machine on the same host using the specified new name and location. In order to perform a clone operation, the original machine must be registered with the Virtuozzo Service. A clone can be created as a virtual machine or as a template. To create a template, set the nFlags parameter to PACF_CLONEVM_TO_TEMPLATE value. The function can also be used to create new virtual machines from templates. In order to that, the hVm parameter must contain a handle to a template, the name and the path parameters must be set to desired values, and the bCreateTemplate parameter must be set to PRL_FALSE (p. 1626).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_CloneWithUuid Function

Clones an existing virtual machine.

Syntax

```
PRL_HANDLE PrlVm_CloneWithUuid(
    PRL_HANDLE hVm,
    PRL_CONST_STR new_vm_name,
    PRL_CONST_STR new_vm_uuid,
    PRL_CONST_STR new_vm_root_path,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

new_vm_name

The name to use for the new virtual machine.

new_vm_uuid

The uuid to use for the new virtual machine.

new_vm_root_path

Name and path of the directory where the new virtual machine should be created. To create the machine in a default directory, pass an empty string.

nFlags

Clone flags (PCVF_CLONE_TO_TEMPLATE, PCVF_CHANGE_SID, PCVF_LINKED_CLONE, PCVF_IMPORT_BOOT_CAMP). Also PACF_NON_INTERACTIVE_MODE can be used to specify non interactive session working mode (remote side won't ask questions during operation progress in this case). Please note that clone operation can take it's own exclusive lock of virtual machine (in change SID or linked clone modes) - so VM shouldn't be exclusively locked with PrlVm_Lock (p. 473) call before clone operation performed.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function creates an exact copy of the specified virtual machine on the same host using the specified new name and location. In order to perform a clone operation, the original machine must be registered with the Virtuozzo Service. A clone can be created as a virtual machine or as a template. To create a template, set the nFlags parameter to PACF_CLONEVM_TO_TEMPLATE value. The function can also be used to create new virtual machines from templates. In order to that, the hVm parameter must contain a handle to a template, the name and the path parameters must be set to desired values, and the bCreateTemplate parameter must be set to PRL_FALSE (p. 1626).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Commit Function

The PrlVm_Commit function is used in modifying a virtual machine configuration.

Syntax

```
PRL_HANDLE PrlVm_Commit(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The PrlVm_BeginEdit (p. 446) and the PrlVm_Commit functions are used to detect collisions with other clients trying to modify the same virtual machine at the same time. The PrlVm_BeginEdit (p. 446) call timestamps the beginning of the editing operation. It does not lock the machine, so other clients can modify the same machine at the same time. When you are done making the changes, you must call the PrlVm_Commit function to save them in the virtual machine configuration. The function will verify that the configuration has not been modified by another client. If a collision is detected, your changes will be rejected.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Example

The following example illustrates how to use PrlVm_BeginEdit (p. 446) and PrlVm_Commit functions.

```
PRL_HANDLE hJobBeginEdit;
PRL_HANDLE hJobCommit;
PRL_RESULT nJobRetCode;

// Timestamp the beginning of the configuration changes operation.
hJobBeginEdit = PrlVm_BeginEdit(hVm);

// Wait for the job to complete.
ret = PrlJob_Wait(hJobBeginEdit, 10000);

// Check for errors.
PrlJob_GetRetCode(hJobBeginEdit, &nJobRetCode);

if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJobBeginEdit);
    return nJobRetCode;
}

// The code modifying configuration parameters goes here...

// Commit the changes when done.
hJobCommit = PrlVm_Commit(hVm);

// Wait for the job to complete.
ret = PrlJob_Wait(hJobCommit, 10000);

// Check the results of the commit operation.
PrlJob_GetRetCode(hJobCommit, &nJobRetCode);

if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Commit error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJobCommit);
    return nJobRetCode;
}
```

PrlVm_CommitEncryption Function

The PrlVm_CommitEncryption function is used in modifying a virtual machine disk encryption configuration.

Syntax

```
PRL_HANDLE PrlVm_CommitEncryption(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags,
    PRL_HANDLE hPolicyList,
    PRL_HANDLE hReserved
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nFlags

A bitset of flags from the PRL_COMMIT_ENCRYPTION_FLAGS (p. 1041) enumeration

hPolicyList

Reserved for future use, its value is ignored.

hReserved

Reserved for future use, its value is ignored.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The PrlVm_BeginEdit (p. 446) and the PrlVm_CommitEncryption functions are used to detect collisions with other clients trying to modify the same virtual machine at the same time. The PrlVm_BeginEdit (p. 446) call timestamps the beginning of the editing operation. It does not lock the machine, so other clients can modify the same machine at the same time. When you are done making the changes, you must call the PrlVm_CommitEncryption function to save them in the virtual machine configuration. The function will verify that the configuration has not been modified by another client. If a collision is detected, your changes will be rejected.

Note that changes in the virtual machine encryption settings can only be committed using this function. Committing encryption changes along with other changes in the virtual machine configuration using the PrlVm_Commit (p. 453) or PrlVm_CommitEx (p. 455) functions will result in error.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_CommitEx Function

The PrlVm_CommitEx function is the same as PrlVm_Commit (p. 453) function but with the flags

Syntax

```
PRL_HANDLE PrlVm_CommitEx(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
```

```
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nFlags

Bitset of flags - from enum PRL_API_COMMAND_FLAGS (p. 1037)

PrlVm_Compact Function

Start process of optimization of virtual hard disk.

Syntax

```
PRL_HANDLE PrlVm_Compact(
    PRL_HANDLE hVm,
    PRL_UINT32 uMask,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

valid PRL_HANDLE (p. 820) describing the virtual machine

uMask

bit mask of really hard disk devices (little endian order). See PRL_HARD_DISK_IFACE_MASK (p. 1065) enumeration for all bits.

nFlags

only bit PACF_NON_INTERACTIVE_MODE is used. Others bits are reserved

Returns

PRL_HANDLE (p. 820) to job that describing async operation result (PRL_INVALID_HANDLE (p. 1627) means not enough memory to instantiate async job represents object handle). Operation return code can be extracted with @ref PrlJob_GetRetCode (p. 143)() call. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle was passed PRL_ERR_UNINITIALIZED (p. 1373) - unexpected error: uninitialized with server connection or non identified VM object

Remarks

The machine should be registered on the server to do that.

PrlVm_Connect Function

Connect to remote desktop.

Syntax

```
PRL_HANDLE PrlVm_Connect(
    PRL_HANDLE hObj,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hObj

A handle of type PHT_VIRTUAL_MACHINE (p. 442) or PHT_DESKTOP_CONTROL (p. 32).

nFlags

Flags, must be 0.

Returns

A handle of type PHT_JOB (p. 138) containing the results of operation.

PrlVm_ConvertDisks Function

Start process conversion of the virtual hard disk(s).

Syntax

```
PRL_HANDLE PrlVm_ConvertDisks(
    PRL_HANDLE hVm,
    PRL_UINT32 uMask,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

valid PRL_HANDLE (p. 820) describing the virtual machine

uMask

bit mask of really hard disk devices (little endian order). See PRL_HARD_DISK_IFACE_MASK (p. 1065) enumeration for all bits.

nFlags

See PRL_CONVERT_VIRTUAL_DISK (p. 1042) enumeration for using bits.

Returns

PRL_HANDLE (p. 820) to job that describing async operation result (PRL_INVALID_HANDLE (p. 1627) means not enough memory to instantiate async job represents object handle). Operation return code can be extracted with @ref PrlJob_GetRetCode (p. 143)() call. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle was passed PRL_ERR_UNINITIALIZED (p. 1373) - unexpected error: uninitialized with server connection or non identified VM object

Remarks

The machine should be registered on the server to do that.

PrlVm_CreateEvent Function

Creates VM event that bint to current VM.

Syntax

```
PRL_RESULT PrlVm_CreateEvent(
    PRL_HANDLE hVm,
    PRL_HANDLE_PTR phEvent
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

unnamed_param

buffer for storing result

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle or non valid (null) pointer to buffer was specified PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to instantiate VM event object PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVm_CreateSnapshot Function

Creates a virtual machine snapshot.

Syntax

```
PRL_HANDLE PrlVm_CreateSnapshot(
    PRL_HANDLE hVm,
    PRL_CONST_STR sName,
    PRL_CONST_STR sDescription
);
```

File

PrlApiVm.h

Parameters

sName

Snapshot name.

sDescription

Snapshot description.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_DeleteSnapshot (p. 461)

PrlVm_GetSnapshotsTreeEx (p. 468)

PrlVm_SwitchToSnapshot (p. 497)

PrlVm_UpdateSnapshotData (p. 503)

PrlVm_CreateSnapshotEx Function

Creates a virtual machine snapshot.

Syntax

```
PRL_HANDLE PrlVm_CreateSnapshotEx(
    PRL_HANDLE hVm,
    PRL_CONST_STR sName,
    PRL_CONST_STR sDescription,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

sName

Snapshot name.

sDescription

Snapshot description.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nFlag

Flags from enum PRL_CREATE_SNAPSHOT_FLAGS (p. 1045)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_DeleteSnapshot (p. 461)

PrlVm_GetSnapshotsTreeEx (p. 468)

PrlVm_SwitchToSnapshot (p. 497)

PrlVm_UpdateSnapshotData (p. 503)

PrlVm_CreateUnattendedFloppy Function

Creates a floppy disk image in the virtual machine home directory for unattended Windows installation.

Syntax

```
PRL_HANDLE PrlVm_CreateUnattendedFloppy(
    PRL_HANDLE hVm,
    PRL_GUEST_DISTRO_VERSION_ENUM nGuestDistroType,
    PRL_CONST_STR sUsername,
    PRL_CONST_STR sCompanyName,
    PRL_CONST_STR sSerialKey
);
```

File

PrlApiVm.h

Parameters

hVm
A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nGuestDistroType
Windows version.

sUsername
User name.

sCompanyName
Company name.

sSerialKey
Serial number.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The following steps describe how to prepare a virtual machine for unattended Windows installation:

1. Create a virtual machine. The machine should have all the necessary devices, like hard drive and CD drive.
2. Execute the PrlVm_CreateUnattendedFloppy function to create a floppy disk image.
3. Insert Windows installation CD-ROM into the drive (or mount a disk image).

4. Start the virtual machine. The machine will boot from the CD.
5. Perform unattended Windows installation as usual.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Delete Function

Deletes the specified virtual machine from the host.

Syntax

```
PRL_HANDLE PrlVm_Delete(
    PRL_HANDLE hVm,
    PRL_HANDLE hDevicesList
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

hDevicesList

A handle of type PHT_STRINGS_LIST (p. 357) containing a list of files (devices) to be deleted from the virtual machine directory. If the list is empty or the parameter contains a null pointer, the entire virtual machine directory will be deleted from the host. If the list contains one or more items, only the specified files will be delete (the rest of the virtual machines files will remain on the host in the original directory).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

This function un-registers the machine from the Virtuozzo Service and deletes all its files. This operation cannot be undone. The specified machine must be registered with the Virtuozzo Service. To un-register a machine from the Virtuozzo Service without deleting any of its file, use the PrlVm_Unreg (p. 500) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_DeleteSnapshot Function

Deleted the specified virtual machine snapshot.

Syntax

```
PRL_HANDLE PrlVm_DeleteSnapshot(
```

```
    PRL_HANDLE hVm,
    PRL_CONST_STR sSnapshotUuid,
    PRL_BOOL bChild
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sSnapshotUuid

Snapshot ID. To obtain a snapshot ID use the PrlVm_GetSnapshotsTreeEx (p. 468) function.

bChild

Specifies whether snapshots derived from the specified snapshot should be deleted or not. True - delete child snapshots. False - don't delete child snapshots.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_CreateSnapshot (p. 458)

PrlVm_GetSnapshotsTreeEx (p. 468)

PrlVm_SwitchToSnapshot (p. 497)

PrlVm_UpdateSnapshotData (p. 503)

PrlVm_Disconnect Function

Disconnect from remote desktop.

Syntax

```
PRL_RESULT PrlVm_Disconnect(
    PRL_HANDLE hObj
);
```

File

PrlApiVm.h

Parameters

hObj

A handle of type PHT_VIRTUAL_MACHINE (p. 442) or PHT_DESKTOP_CONTROL (p. 32).

PrlVm_DropSuspendedState Function

Resets a suspended virtual machine.

Syntax

```
PRL_HANDLE PrlVm_DropSuspendedState(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

When a virtual machine is suspended with the PrlVm_Suspend (p. 496) call, the current state of the virtual machine processes is saved into a temporary file on the hard disk, and the virtual machine is stopped. When a suspended virtual machine is resumed with the PrlVm_Resume (p. 489) call, the data in the temp file is used to restore the state of the virtual machine processes. The PrlVm_DropSuspendedState function resets a suspended virtual machine by deleting the temp file from the drive Once the function successfully completes, the virtual machine can no longer be resumed with the PrlVm_Resume (p. 489) call. It still can be started normally, however.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_GenerateVmDevFilename Function

Automatically generates a name for a virtual device which will be unique within a home directory of the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_GenerateVmDevFilename(
    PRL_HANDLE hVm,
    PRL_CONST_STR sFilenamePrefix,
    PRL_CONST_STR sFilenameSuffix,
    PRL_CONST_STR sIndexDelimiter
);
```

File

PrlApiVm.h

Parameters

sFilenamePrefix

A prefix to add to the name. Pass a null pointer to use a default value: "tmpfile". The name will be generated by adding a number to the prefix (e.g. tmpfile1, tmpfile2, tmpfile3, etc.)

sFilenameSuffix

A suffix to add to the name. Pass a null pointer for no suffix.

sIndexDelimiter

A character(s) that will be inserted between the prefix and the index (e.g. tmpfile_1, tmpfile_2, tmpfile_3, etc). You can specify a null pointer or an empty string if you don't want to insert any extra characters there.

hVM

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the home directory information is missing in the specified virtual machine object. If this is a new virtual machine, try updating the local object by calling the PrlVm_RefreshConfig (p. 481) or PrlSrv_GetVmList (p. 264) function.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_GetConfig Function

Obtains a handle of type PHT_VM_CONFIGURATION (p. 537).

Syntax

```
PRL_RESULT PrlVm_GetConfig(
    PRL_HANDLE hVm,
    PRL_HANDLE_PTR phVmCfg
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

phVmCfg

[out] A pointer to a variable that receives the handle of type PHT_VM_CONFIGURATION (p. 537).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_GetPackedProblemReport Function

The PrlVm_GetPackedProblemReport function obtains a Packed problem report on abnormal virtual machine termination.

Syntax

```
PRL_HANDLE PrlVm_GetPackedProblemReport(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine. PRL_UINT32 (p. 798) nFlags - reserved flags parameter

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

If a virtual machine stops running abnormally, you call this function to obtain a report from the Virtuozzo Service describing the problem.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamAsString (p. 213) function to obtain a string containing path to the report directory.

PrlVm_GetPerfStats Function

Obtains a handle of type PHT_EVENT (p. 72) containing the virtual machine performance statistics.

Syntax

```
PRL_HANDLE PrlVm_GetPerfStats(
    PRL_HANDLE hVm,
    PRL_CONST_STR sFilter
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sFilter

String representing statistics filter, use char '*' at the end to get subset of statistic. Could be NULL, to get all statistic.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the result from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_EVENT (p. 72).

PrlVm_GetProblemReport Function

The PrlVm_GetProblemReport function obtains a problem report on abnormal virtual machine termination.

Syntax

```
PRL_HANDLE PrlVm_GetProblemReport(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

If a virtual machine stops running abnormally, you call this function to obtain a report from the Virtuozzo Service describing the problem.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParamAsString (p. 213) function to obtain a string containing the report.

PrlVm_GetQuestions Function

The PrlVm_GetQuestions function allows to synchronously receive questions from a Virtuozzo Service.

Syntax

```
PRL_RESULT PrlVm_GetQuestions(
    PRL_HANDLE hVm,
    PRL_HANDLE_PTR pQuestionsList
);
```

File

PrlApiVm.h

Parameters

pQuestionsList

[out] A handle of type PHT_HANDLES_LIST (p. 108) containing a list of available questions.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442).

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle or non valid (NULL) pointer to the buffer for storing result was passed. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

It can be used as an alternative to asynchronous question processing through events. You can use this function at any time to check if a question is awaiting the client response on the Virtuozzo Service side.

PrlVm_GetSnapshotsTree Function

Retrieves snapshot information for the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_GetSnapshotsTree(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

flags

A flags from enum PRL_GET_SNAPSHOT_TREE_FLAGS (p. 1061) and PRL_API_COMMAND_FLAGS (p. 1037)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Snapshot information is returned as an XML file containing the snapshot tree. Some of the important XML elements are:

SavedStateItem -- the root element of the branch containing information about an individual snapshot. The element has an attribute "guid" containing the snapshot ID. The ID is required when performing other operations on a snapshot.

Name -- contains the snapshot name.

DateTime -- the snapshot creation date and time.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get a snapshot tree from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamAsString (p. 213) function to obtain a string value containing the snapshots tree XML data.

Please note that no parser is provided to traverse the returned XML. To use this functionality, you will have to parse the returned XML yourself. This function will be replaced in the future releases of the Virtuozzo SDK.

See Also

PrlVm_CreateSnapshot (p. 458)

PrlVm_DeleteSnapshot (p. 461)

PrlVm_SwitchToSnapshot (p. 497)

PrlVm_UpdateSnapshotData (p. 503)

PrlVm_GetSnapshotsTreeEx Function

Syntax

```
PRL_HANDLE PrlVm_GetSnapshotsTreeEx(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

PrlVm_GetState Function

Obtains a handle of type PHT_VM_INFO (p. 719) containing the specified virtual machine state and access rights information.

Syntax

```
PRL_HANDLE PrlVm_GetState(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParam (p. 210) function to obtain the PHT_VM_INFO (p. 719) handle.

Example

```
hJob =
PrlVm_GetState( hVm );
PrlJob_Wait( hJob, 10000 );
PrlJob_GetResult( hJob, &hJobResult );
PrlHandle_Free( hJob );

// Get the handle from job
PRL_HANDLE hHandle;
PrlResult_GetParam( hJobResult, &hHandle );
```

PrlVm_GetStatistics Function

Obtains a handle of type PHT_SYSTEM_STATISTICS (p. 360) containing the virtual machine resource usage information, including CPU(s), memory, disk drives, processes, user session, system uptime, network packets, etc.

Syntax

```
PRL_HANDLE PrlVm_GetStatistics(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.

Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_SYSTEM_STATISTICS (p. 360) containing the resource usage data.

PrlVm_GetStatisticsEx Function

The PrlVm_GetStatisticsEx function is the same as PrlVm_GetStatistics (p. 469) function but with the flags

Syntax

```
PRL_HANDLE PrlVm_GetStatisticsEx(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nFlags

Bitset of flags - from enum PRL_API_COMMAND_FLAGS (p. 1037)

PrlVm_GetSuspendedScreen Function

Retrieves the virtual machine screen state before it was suspending.

Syntax

```
PRL_HANDLE PrlVm_GetSuspendedScreen(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_VM_IS_NOT_SUSPENDED (p. 1403) - the virtual machine is not in the suspended state.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_GetToolsState Function

Obtains a handle of type PHT_VM_TOOLS_INFO (p. 724) containing information about the state of the Virtuozzo Tools package in a virtual machine.

Syntax

```
PRL_HANDLE PrlVm_GetToolsState(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Example

The following example illustrates how to obtain a handle of type PHT_VM_TOOLS_INFO (p. 724). The hVm variable is a handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

```
PRL_HANDLE hJob, hResult, hVmToolsInfo;

hJob = PrlVm_GetToolsState( hVm );
PrlJob_Wait( hJob, 1000 );

PrlJob_GetResult( hJob, &hResult );
PrlHandle_Free( hJob );

PrlResult_GetParam( hJobResult, &hVmToolsInfo );
PrlHandle_Free( hJobResult );
```

PrlVm_InitiateDevStateNotifications Function

This method lets to initiate sending notifications about devices states by VM.

Syntax

```
PRL_HANDLE PrlVm_InitiateDevStateNotifications(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_HANDLE (p. 820) to job that describing async operation result (PRL_INVALID_HANDLE (p. 1627) means not enough memory to instantiate async job represents object handle). Operation return code can be extracted with @ref PrlJob_GetRetCode (p. 143)() call. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle was passed PRL_ERR_UNINITIALIZED (p. 1373) - unexpected error: uninitialized with the Virtuozzo Service connection or non identified VM object

Remarks

Those notifications have to be same as notifications after changing a device state

PrlVm_InstallTools Function

Begin the Virtuozzo Tools installation process.

Syntax

```
PRL_HANDLE PrlVm_InstallTools(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_HANDLE (p. 820) to job that describing async operation result (PRL_INVALID_HANDLE (p. 1627) means not enough memory to instantiate async job represents object handle). Operation return code can be extracted with @ref PrlJob_GetRetCode (p. 143)() call. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid PHT_VIRTUAL_MACHINE (p. 442) handle was passed PRL_ERR_UNINITIALIZED (p. 1373) - unexpected error: uninitialized with Virtuozzo Service connection or non identified VM object

Remarks

The machine must be registered with the Virtuozzo Service to perform this operation.

PrlVm_Lock Function

Lets to exclusively lock VM for current session.

Syntax

```
PRL_HANDLE PrlVm_Lock(
    PRL_HANDLE hVm,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

nReserved

reserved param for further purposes

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

After this call just current sesiion be able to operate with locked VM until PrlVm_Unlock (p. 499) will be called or current session will be closed. If VM was already locked by some another session (or the same one) then lock operation will fail.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are: PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_LoginInGuest Function

Creates a new console session or binds to an existing GUI session in a virtual machine.

Syntax

```
PRL_HANDLE PrlVm_LoginInGuest(
    PRL_HANDLE hVm,
    PRL_CONST_STR sUserName,
    PRL_CONST_STR sUserPassword,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sUserName

User name. This parameter must contain a valid guest OS user name or one of the following constants:

PRL_CURRENT_GUEST_OS_SESSION (p. 1624) -- binds to an existing GUI session in a virtual machine.

PRL_PRIVILEGED_GUEST_OS_SESSION (p. 1632) -- binds to an existing session under which Virtuozzo Service is running. This session belongs to a privileged user (e.g. root on Linux or LocalSystem on Windows), so it can be used to perform administrative tasks.

Note that in order to use the constants above, the current user must be an administrator of the host OS or be the owner of the target virtual machine.

sUserPassword

User password. If the sUserName parameter contains one of the constants (see above) then this parameter can contain an empty string.

nFlags

A bitset of flags. Must be 0 in the current API implementation.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Note: To use this function, Virtuozzo Tools must be installed in the target virtual machine.

This function is used when you want to run a program in a virtual machine or perform an administrative task that require native guest operating system login. Depending on the type of the program that you would like to run, the login must be performed using one of the following methods:

- To run a GUI application, call this function passing the PRL_CURRENT_GUEST_OS_SESSION (p. 1624) constant in the sUserName parameter. The reason is the function can only bind to an existing GUI session and cannot create a new one. Prior to calling this function, a GUI session must already be open in a virtual machine (the machine must be running and a user must be logged in to it). The user password parameter can be blank.
- To run a console program, the sUserName parameter can contain the PRL_PRIVILEGED_GUEST_OS_SESSION (p. 1632) constant or a valid user name and password. When using PRL_PRIVILEGED_GUEST_OS_SESSION (p. 1632), the function will bind to an existing session under which Virtuozzo Service is running. This session belongs to a privileged user (e.g. root on Linux or LocalSystem on Windows), so it can be used to perform administrative tasks. When specifying a user name and password, a new session will be created in the virtual machine.

On successful login, the function obtains a handle of type PHT_VM_GUEST_SESSION (p. 716) (see below for the instructions on how to get the handle from the PHT_JOB (p. 138) object). You can use the handle functions to run programs in a virtual machine and to perform other supported operations. See the PHT_VM_GUEST_SESSION (p. 716) handle description for a complete list of functions.

When the session is no longer needed, it should be closed using the PrlVmGuest_Logout (p. 717) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the PHT_VM_GUEST_SESSION (p. 716) handle from the returned PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain the PHT_RESULT (p. 210) handle.

2. Use the PrlResult_GetParam (p. 210) to obtain the PHT_VM_GUEST_SESSION (p. 716) handle.

See Also

PrlVmGuest_RunProgram (p. 718)

PrlVmGuest_Logout (p. 717)

PrlVm_Migrate Function

Migrates an existing virtual machine to another host.

Syntax

```
PRL_HANDLE PrlVm_Migrate(
    PRL_HANDLE hVm,
    PRL_HANDLE hTargetServer,
    PRL_CONST_STR target_home_path,
    PRL_UINT32 migration_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

hTargetServer

A handle of type PHT_SERVER (p. 219) identifying the target Virtuozzo Service.

target_home_path

Target VM home path (i.e. where VM will be placed on target system). Empty string accounts as default VM directory path

migration_flags

Flags that specify migration type and etc. See acceptable flags at PrlCommandsFlags.h

reserved_flags

Reserved parameter for further purposes

force_operation

Specifies to process VM migration action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_MigrateCancel Function

Cancels the migration operation that was started with PrlVm_Migrate (p. 475) or PrlVm_MigrateEx (p. 476).

Syntax

```
PRL_HANDLE PrlVm_MigrateCancel(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_MigrateEx Function

Migrates an existing virtual machine to another host.

Syntax

```
PRL_HANDLE PrlVm_MigrateEx(
    PRL_HANDLE hVm,
    PRL_CONST_STR target_host,
    PRL_UINT32 target_port,
    PRL_CONST_STR target_session_id,
    PRL_CONST_STR target_home_path,
    PRL_UINT32 migration_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

target_host

The name of the target host machine.

target_port

The port number on the target host.

target_session_id

The target Virtuozzo Service session ID. This ID can be retrieved with PrlLoginResponse_GetSessionUuid (p. 153) from the response of the login request.

target_home_path

Target VM home path (i.e. where VM will be placed on target system). Empty string accounts as default VM directory path

migration_flags

Flags that specify migration type and etc. See acceptable flags at PrlCommandsFlags.h

reserved_flags

Reserved parameter for further purposes

force_operation

Specifies to process VM migration action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

This is an extended version of PrlVm_Migrate (p. 475) function that allows to initiate VM migration process without active connection to target Virtuozzo Service.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_MigrateWithRename Function

Migrates an existing virtual machine to another host.

Syntax

```
PRL_HANDLE PrlVm_MigrateWithRename(
    PRL_HANDLE hVm,
    PRL_HANDLE hTargetServer,
    PRL_CONST_STR target_name,
    PRL_CONST_STR target_home_path,
    PRL_UINT32 migration_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

hTargetServer

A handle of type PHT_SERVER (p. 219) identifying the target Virtuozzo Service.

target_name

Target VM name (i.e. what name will be given to VM on target system). Specify empty string to preserve VM name from source system.

target_home_path

Target VM home path (i.e. where VM will be placed on target system). Empty string accounts as default VM directory path

migration_flags

Flags that specify migration type and etc. See acceptable flags at PrlCommandsFlags.h

reserved_flags

Reserved parameter for further purposes

force_operation

Specifies to process VM migration action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_MigrateWithRenameEx Function

Migrates an existing virtual machine to another host.

Syntax

```
PRL_HANDLE PrlVm_MigrateWithRenameEx(
    PRL_HANDLE hVm,
    PRL_CONST_STR target_host,
    PRL_UINT32 target_port,
    PRL_CONST_STR target_session_id,
    PRL_CONST_STR target_name,
    PRL_CONST_STR target_home_path,
    PRL_UINT32 migration_flags,
    PRL_UINT32 reserved_flags,
    PRL_BOOL force_operation
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

target_host

The name of the target host machine.

target_port

The port number on the target host.

target_session_id

The target Virtuozzo Service session ID. This ID can be retrieved with PrlLoginResponse_GetSessionUuid (p. 153) from the response of the login request.

target_home_path

Target VM home path (i.e. where VM will be placed on target system). Empty string accounts as default VM directory path

migration_flags

Flags that specify migration type and etc. See acceptable flags at PrlCommandsFlags.h

reserved_flags

Reserved parameter for further purposes

force_operation

Specifies to process VM migration action without asking any additional questions from the Virtuozzo Service side (non-interactive clients should use this option).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

This is an extended version of PrlVm_MigrateWithRename (p. 477) function that allows to initiate VM migration process without active connection to target Virtuozzo Service.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Mount Function

Mount the specified virtual machine Virtual Disks.

Syntax

```
PRL_HANDLE PrlVm_Mount(
    PRL_HANDLE hVm,
    PRL_CONST_STR sMntPath,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sMntPath

not used

nFlags

Mount flags of type PRL_MOUNT_VIRTUAL_DISK (p. 1079)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Move Function

Move bundle of virtual machine or private area of container.

Syntax

```
PRL_HANDLE PrlVm_Move(
    PRL_HANDLE hVm,
    PRL_CONST_STR sNewHomePath,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine or container.

sNewHomePath

Target path for Vm bundle or CT private

nFlags

Flags that specify move type. Not used yet.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The function move all content of bundle to specify location. This function will move only stopped or suspended Vm/Ct.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Pause Function

Pauses the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Pause(
    PRL_HANDLE hVm,
    PRL_BOOL bAcpi
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

bAcpi

Specifies if ACPI should be used to pause the virtual machine. If set to PRL_FALSE (p. 1626), ACPI will not be used. In this case, all processes of the virtual machine will simply be stopped. If the parameter is set to PRL_TRUE (p. 1652), ACPI will be used to pause the machine. With ACPI, the machine will be paused using a method supported by the guest operating system (sleep, hibernate, etc.). The actual method used depends on the type and configuration of the guest OS.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

A paused virtual machine can be resumed using the PrlVm_Start (p. 491) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_Start (p. 491)

PrlVm_RefreshConfig Function

The PrlVm_RefreshConfig function updates the configuration information contained in the specified virtual machine object (handle).

Syntax

```
PRL_HANDLE PrlVm_RefreshConfig(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine. This is the object that will be updated.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

When you obtain a virtual machine handle that refers to an existing virtual machine, it is populated with the virtual machine configuration data. When working with configuration data, you always work with this local copy. Naturally, the configuration can change on the Virtuozzo Service side at any time. The PrlVm_RefreshConfig function allows to re-retrieve the configuration data from the Virtuozzo Service. The function can be used when an event of type PET_DSP_EVT_VM_CONFIG_CHANGED is triggered (the event notifies the client that the virtual machine configuration has been modified), or on unsuccessful PrlVm_Commit (p. 453) operation, or at any time you want to synchronize your local virtual machine object with the Virtuozzo Service. Please note that the function replaces the local data, so if you made any modification to it, the changes will be lost.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_RefreshConfigEx Function

The PrlVm_RefreshConfigEx function updates the configuration information contained in the specified virtual machine object (handle).

Syntax

```
PRL_HANDLE PrlVm_RefreshConfigEx(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine. This is the object that will be updated.

nFlags

A bitset of flags.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

When you obtain a virtual machine handle that refers to an existing virtual machine, it is populated with the virtual machine configuration data. When working with configuration data, you always work with this local copy. Naturally, the configuration can change on the Virtuozzo Service side at any time. The PrlVm_RefreshConfigEx function allows to re-retrieve the configuration

data from the Virtuozzo Service. The function can be used when an event of type PET_DSP_EVT_VM_CONFIG_CHANGED is triggered (the event notifies the client that the virtual machine configuration has been modified), or on unsuccessful PrlVm_Commit (p. 453) operation, or at any time you want to synchronize your local virtual machine object with the Virtuozzo Service. Please note that the function replaces the local data, so if you made any modification to it, the changes will be lost.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Reg Function

See also PrlVm_RegEx (p. 485)

Syntax

```
PRL_HANDLE PrlVm_Reg(
    PRL_HANDLE hVm,
    PRL_CONST_STR sVmParentPath,
    PRL_BOOL bNonInteractiveMode
);
```

File

PrlApiVm.h

Example

The following example shows how to create a new virtual machine. The sample assumes that the client program has already obtained a server object handle (hServer) and performed the Virtuozzo Service login operation.

```
PRL_HANDLE hVm;
PRL_HANDLE hResult;
PRL_RESULT nJobRetCode;
PRL_RESULT ret;

// Obtain a new virtual machine handle.
ret = PrlSrv_CreateVm(hServer, &hVm);
if (PRL_FAILED(ret))
{
    // Error handling goes here...
    return ret;
}

// Get the host config info.
hJob = PrlSrv_GetSrvConfig(hServer);
ret = PrlJob_Wait(hJob, 10000);

// Check the return code of PrlSrv_GetSrvConfig.
PrlJob_GetRetCode(hJob, &nJobRetCode);
if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hVm);
    return nJobRetCode;
}

// Get a handle to the object containing the result of PrlSrv_GetSrvConfig,
// and then get a handle to the server configuration object from it.
ret = PrlJob_GetResult(hJob, &hResult);
```

```
PRL_HANDLE hSrvCfg;
PrlResult_GetParam(hResult, &hSrvCfg);

// Delete the job and the result handles.
PrlHandle_Free(hJob);
PrlHandle_Free(hResult);

// Now that we have the host machine configuration data,
// we can set the default configuration for the new virtual machine.
ret = PrlVm_SetDefaultConfig(
                hVm, // VM handle.
                hSrvCfg, // Host config data.
                PVS_GUEST_TYPE_WINDOWS, // Target OS type.
                PVS_GUEST_VER_WIN_2003, // Target OS version.
                PRL_TRUE); // Create and connect devices.

if (PRL_FAILED(ret))
{
    fprintf(stderr, "Error: %s\n", PRL_RESULT_TO_STRING(ret));
    PrlHandle_Free(hSrvCfg);
    PrlHandle_Free(hVm);
    return ret;
}

PrlHandle_Free(hSrvCfg);

// Set the virtual machine name.
ret = PrlVm_SetName(hVm, "My Windows Server 2003");

// The following two calls demonstrate how to modify
// some of the default values of the virtual machine configuration.
// These calls are optional. You may remove them to use the default values.

// Set the RAM size for the machine to 256 MB.
ret = PrlVm_SetRamSize(hVm, 256);

// Set the virtual hard disk size to 20 GB.
// First, get the handle to the hard disk object using the
// PrlVm_GetHardDisk function. The index of 0 is used
// because the default configuration has just one virtual hard disk.
// After that, use the handle to set the disk size to the desired value.

PRL_HANDLE hHDD;
ret = PrlVm_GetHardDisk(hVm, 0, &hHDD);
ret = PrlVmDevHd_SetDiskSize(hHDD, 20000);

// Create and register the machine with the Virtuozzo Service.
hJob = PrlVm_Reg(hVm, // VM handle.
                 "", // VM root directory (using default).
                 PRL_TRUE); // Using non-interactive mode.

// Wait for the operation to complete.
ret = PrlJob_Wait(hJob, 10000);

// Check the return code of PrlVm_Reg.
PrlJob_GetRetCode(hJob, &nJobRetCode);
if (PRL_FAILED(nJobRetCode))
{
    fprintf(stderr, "Error: %s\n", PRL_RESULT_TO_STRING(nJobRetCode));
    PrlHandle_Free(hJob);
    PrlHandle_Free(hVm);
    return nJobRetCode;
}

// Delete handles.
PrlHandle_Free(hJob);
```

```
PrlHandle_Free(hVm);
```

PrlVm_RegEventHandler Function

Registers an event handler (callback function) with the virtual machine.

Syntax

```
PRL_RESULT PrlVm_RegEventHandler(
    PRL_HANDLE hObj,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR userData
);
```

File

PrlApiVm.h

Parameters

hObj

A handle of type PHT_VIRTUAL_MACHINE (p. 442) or PHT_DESKTOP_CONTROL (p. 32).

handler

A pointer to the callback function that will handle events.

userData

A pointer to a user data that will be passed to the callback function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

To use the callback functionality, you first have to implement a callback function. The callback function must have a specific signature (see PRL_METHOD_PTR (p. 1631) for the prototype). After you register an event handler with a virtual machine, the callback function will be called every time an event is generated in the machine.

You can also register an event handler with a Virtuozzo Service (instead of a particular virtual machine) using the PrlSrv_RegEventHandler (p. 284) function. If you register the event handler with the Virtuozzo Service, you will be able to receive events from all of the virtual machines registered with that Virtuozzo Service. For more information on events and asynchronous functions, see the Virtuozzo Virtualization SDK Programmer's Guide.

PrlVm_RegEx Function

Creates a new virtual machine and registers it with the Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlVm_RegEx(
    PRL_HANDLE hVm,
    PRL_CONST_STR sVmParentPath,
    PRL_UINT32 nFlags
```

```
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sVmParentPath

Optional. Name and path of the parent directory where the virtual machine directory should be created. If this parameter is omitted (null pointer or empty string is passed), the new virtual machine directory will be created in the default directory for this Virtuozzo Service.

nFlags

Flags set which specifies way of virtual machine creation. Use PACF_NON_INTERACTIVE_MODE flag here instead of deprecated bNonInteractiveMode parameter. Other bitset of flags - from enum PRL_CREATE_DISK_IMAGE_FLAGS (p. 1045)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The steps involved in creating a typical virtual machine are:

1. Obtain a handle to a new virtual machine object using the PrlSrv_CreateVm (p. 232) function.
2. Populate the object with the desired configuration data.
3. Choose a name for the virtual machine and set it using the PrlVmCfg_SetName (p. 682) function.
4. Create and register the new machine with a Virtuozzo Service using the PrlVm_Reg (p. 483) function. This step will create the necessary virtual machine files on the host and will add the machine to the Virtuozzo Service VM registry. The directory containing the virtual machine files will have the same name as the name you've chosen for your virtual machine. The directory will be created in the default VM root directory for this Virtuozzo Service. If you would like to create the virtual machine directory in a different location, you may specify the desired parent directory name and path.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Reset Function

Resets the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Reset(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

This function is an equivalent of PrlVm_Stop (p. 493) and PrlVm_Start (p. 491) functions executed in succession. The stop operation will NOT use ACPI, so the entire reset operation will resemble the "Reset" button pressed on a physical box.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_ResetUptime Function

Resets the virtual machine uptime counter (start date and counter itself).

Syntax

```
PRL_HANDLE PrlVm_ResetUptime(
    PRL_HANDLE hVm,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nFlags

Reserved for future use.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object. To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are: PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully. PRL_ERR_UNINITIALIZED (p. 1373) ??“ unknown PHT_VIRTUAL_MACHINE (p. 442) handle or invalid binding to server connection.

PrlVm_Restart Function

Restarts the operating system of the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Restart(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or start mode of virtual machine.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Restore Function

Restores the registered virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Restore(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation, including the return code and a handle of type PHT_VIRTUAL_MACHINE (p. 442) containing information about the virtual machine that was registered.

PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the PHT_JOB (p. 138) object:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_VIRTUAL_MACHINE (p. 442).

PrlVm_Resume Function

Resumes a virtual machine that was previously suspended with the PrlVm_Suspend (p. 496) call.

Syntax

```
PRL_HANDLE PrlVm_Resume(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_Suspend (p. 496)

PrlVm_SetConfig Function

The PrlVm_SetConfig is reserved and does nothing.

Syntax

```
PRL_RESULT PrlVm_SetConfig(
    PRL_HANDLE hVm,
    PRL_HANDLE hVmCfg
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_SetUserPasswd Function

Allows to change the password of a guest operating system user.

Syntax

```
PRL_HANDLE PrlVm_SetUserPasswd(
    PRL_HANDLE hVm,
    PRL_CONST_STR sUserName,
    PRL_CONST_STR sUserPasswd,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sUserName

Guest operating system user name. Can be null or empty string when setting the administrator password.

sUserPasswd

New password. Can be null when setting an empty password.

nFlags

Bitset of flags - from enum PRL_SET_USERPASSWD_FLAGS (p. 1092)

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Start Function

Start the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Start(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.s

Remarks

This function can be used to start a stopped or paused virtual machine.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_Pause (p. 481)

PrlVm_StartEx Function

Starts the specified virtual machine using the specified mode (see the PRL_VM_START_MODE (p. 950) enumeration).

Syntax

```
PRL_HANDLE PrlVm_StartEx(
    PRL_HANDLE hVm,
    PRL_UINT32 nStartMode,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nStartMode

Virtual machine start mode.

nReserved

This is a reserved parameter.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Use this function instead of the PrlVm_Start (p. 491) function to start the virtual machine in a normal or safe mode. Other modes may be added in the future.

If VM is starting in PSM_VM_START_FOR_COMPACT mode then flags (third argument) have to be equal PRL_HARD_DISK_IFACE_MASK (p. 1065) enum.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or start mode of virtual machine.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_StartVncServer Function

Starts a VNC server for the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_StartVncServer(
    PRL_HANDLE hVm,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nReserved

Reserved parameter for future purposes. Must contain 0 in the current implementation.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer for user name was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_StopVncServer (p. 494)

PrlVm_Stop Function

Stop the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Stop(
    PRL_HANDLE hVm,
    PRL_BOOL bGraceful
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

bGraceful

If graceful shutdown is used, the machine will be properly shutdown by means of Guest tools (if available), or by ACPI request, If shutdown timeout expired the Vm will be forcebly stopped. If graceful shutdown is not used, the machine will be simply "unplugged"

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_StopEx Function

Stop the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_StopEx(
    PRL_HANDLE hVm,
    PRL_UINT32 nStopMode,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nStopMode

Virtual machine stop mode, an enumerator from the PRL_VM_STOP_MODE (p. 951) enumeration.

nFlags

Virtual machine stop option, an enumerator from the PRL_VM_STOP_FLAGS (p. 950) enumeration.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_StopVncServer Function

Stops the VNC server that was previously started for a virtual machine.

Syntax

```
PRL_HANDLE PrlVm_StopVncServer(
    PRL_HANDLE hVm,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nReserved

Reserved parameter for future purposes. Must be 0 in the current implementation.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer for user name was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_StartVncServer (p. 492)

PrlVm_SubscribeToGuestStatistics Function

Subscribes to receive the specified virtual machine performance statistics on a periodic basis.

Syntax

```
PRL_HANDLE PrlVm_SubscribeToGuestStatistics(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To obtain statistics, an event handler (callback function) is required. Within the event handler, first check the type of event. Events of type PET_DSP_EVT_VM_STATISTICS_UPDATED indicate an event containing performance data. To access the statistics handle (a handle of type PHT_SYSTEM_STATISTICS (p. 360)), first extract the event parameter using PrlEvent_GetParam (p. 84), then convert the result (which will be a handle to an object of type PHT_EVENT_PARAMETER (p. 88)) to a handle using PrlEvtPrm_ToHandle (p. 92). The functions that operate on PHT_SYSTEM_STATISTICS (p. 360) references can then be used to obtain statistics data.

For the event handler to be called, it is necessary to register it with PrlVm_RegEventHandler (p. 485). When statistics data is no longer required, unsubscribe from statistics events using PrlVm_UnsubscribeFromGuestStatistics (p. 501). When events are no longer required, unregister the event handler using PrlVm_UnregEventHandler (p. 500).

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_SubscribeToHostStatistics (p. 301)

PrlVm_SubscribeToPerfStats Function

Allows to subscribe to periodicaly receive virtual machine perfomance statistics.

Syntax

```
PRL_HANDLE PrlVm_SubscribeToPerfStats(
    PRL_HANDLE hVm,
    PRL_CONST_STR sFilter
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

sFilter

String representing statistics filter, use char '*' at the end to get subset of statistic. Could be NULL, to get all statistic.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To obtain statistics, an event handler (callback function) is required. Within the event handler, first check the type of event. Events of type PET_DSP_EVT_VM_PERFSTATS indicate an event containing statistics data. The perfomace statistics repesented by parameters of Event object (a handle of type PHT_EVENT (p. 72)).

For the event handler to be called, it is necessary to register it with PrlSrv_RegEventHandler (p. 284). When statistics data is no longer required, unsubscribe from statistics events using PrlVm_UnsubscribeFromHostPerfStats. When events are no longer required, unregister the event handler using PrlVm_UnregEventHandler (p. 500).

If method is called mutliple times, the 'filter' value is updated each time on the Virtuozzo Service side.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Suspend Function

Suspends the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_Suspend(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

When a running virtual machine is suspended, the state of the virtual machine processes is saved into a file on the host and

after that the machine is stopped. To resume the machine, use the PrlVm_Resume (p. 489) function. If, after suspending a machine, you don't want to resume it but would like to return to the previous state, you may call the PrlVm_DropSuspendedState (p. 463) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_Resume (p. 489)

PrlVm_DropSuspendedState (p. 463)

PrlVm_SwitchToSnapshot Function

Reverts the specified virtual machine to the specified snapshot.

Syntax

```
PRL_HANDLE PrlVm_SwitchToSnapshot(
    PRL_HANDLE hVm,
    PRL_CONST_STR sSnapshotUuid
);
```

File

PrlApiVm.h

Parameters

sSnapshotUuid

The ID of a snapshot to revert to. To obtain a list of the available snapshots and their IDs, use the PrlVm_GetSnapshotsTreeEx (p. 468) function.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_CreateSnapshot (p. 458)

PrlVm_DeleteSnapshot (p. 461)

PrlVm_GetSnapshotsTreeEx (p. 468)

PrlVm_UpdateSnapshotData (p. 503)

PrlVm_SwitchToSnapshotEx Function

Reverts the specified virtual machine to the specified snapshot.

Syntax

```
PRL_HANDLE PrlVm_SwitchToSnapshotEx(
    PRL_HANDLE hVm,
    PRL_CONST_STR sSnapshotUuid,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

sSnapshotUuid

The ID of a snapshot to revert to. To obtain a list of the available snapshots and their IDs, use the PrlVm_GetSnapshotsTreeEx (p. 468) function.

nFlags

Bitset of flags - from enum PRL_SWITCH_SNAPSHOT_FLAGS (p. 1094)

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_CreateSnapshot (p. 458)

PrlVm_DeleteSnapshot (p. 461)

PrlVm_GetSnapshotsTreeEx (p. 468)

PrlVm_UpdateSnapshotData (p. 503)

PrlVm_Umount Function

Unmount the specified virtual machine Virtual Disks.

Syntax

```
PRL_HANDLE PrlVm_Umount(
    PRL_HANDLE hVm,
```

```
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Unlock Function

Lets to unlock previously locked with PrlVm_Lock (p. 473) call VM.

Syntax

```
PRL_HANDLE PrlVm_Unlock(
    PRL_HANDLE hVm,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

nReserved

reserved param for further purposes

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Note that just VM lock owner session can unlock VM.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_Unreg Function

Unregisters the specified virtual machine from the Virtuozzo Service.

Syntax

```
PRL_HANDLE PrlVm_Unreg(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

This call removes the virtual machine from the Virtuozzo Service registry, but does not delete the virtual machine files from the hard drive. You can re-register a machine with the Virtuozzo Service at any time using the PrlSrv_RegisterVm (p. 286) or PrlSrv_RegisterVmEx (p. 288) functions call.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_UnregEventHandler Function

Unregisters an event handler (callback function) that was previously registered using the PrlVm_RegEventHandler (p. 485) function.

Syntax

```
PRL_RESULT PrlVm_UnregEventHandler(
    PRL_HANDLE hObj,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR userData
);
```

File

PrlApiVm.h

Parameters

hObj

A handle of type PHT_VIRTUAL_MACHINE (p. 442) or PHT_DESKTOP_CONTROL (p. 32).

handler

A pointer to the callback function. This must be the same pointer that was passed to the PrlVm_RegEventHandler (p. 485) when the event handler was registered with the virtual machine.

userData

A pointer to the user data. This must also be the same pointer that was passed to the PrlVm_RegEventHandler (p. 485) when the event handler was registered with the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use this function when you no longer want to receive event notifications from the virtual machine. The values of the handler and userData parameter constitute a unique pair which will be used to identify the event handler that will be unregistered. Make sure that you supply the same values as the ones you used when you registered the handler.

PrlVm_UnsubscribeFromGuestStatistics Function

Cancels the performance statistics subscription that was previously initiated using the PrlVm_SubscribeToGuestStatistics (p. 495) function.

Syntax

```
PRL_HANDLE PrlVm_UnsubscribeFromGuestStatistics(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_UnsubscribeFromPerfStats Function

Cancels the Virtuozzo Service performance statistics subscription that was started using the PrlVm_SubscribeToPerfStats (p. 495) function.

Syntax

```
PRL_HANDLE PrlVm_UnsubscribeFromPerfStats(
    PRL_HANDLE hVm
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_UpdateSecurity Function

Updates access level for the specified virtual machine.

Syntax

```
PRL_HANDLE PrlVm_UpdateSecurity(
    PRL_HANDLE hVm,
    PRL_HANDLE hAccessRights
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

hAccessRights

A handle of type PHT_ACCESS_RIGHTS (p. 9) containing the virtual machine access level information.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

By default, only the owner of a virtual machine has full access to it. All other users are completely unaware of the machine (it will not even appear in the list of the available virtual machines for them). The owner of the virtual machine can change that by setting one of the predefined access levels for the machine. Depending on the level of access set by the owner, other users will be able to see the machine and to perform tasks on it.

To update the access level, obtain a handle of type PHT_ACCESS_RIGHTS (p. 9), set the desired level using the PrlAcl_SetAccessForOthers (p. 12) function, and then call this function to apply the changes to the virtual machine. The caller must be the owner of the machine to perform this operation. The PrlAcl_IsCurrentSessionOwner (p. 12) function can be used to determine if the current user is the owner of the virtual machine.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_ACCESS_DENIED (p. 1175) - current user is not the owner of the target virtual machine.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVm_UpdateSnapshotData Function

Modifies the specified virtual machine snapshot name and description.

Syntax

```
PRL_HANDLE PrlVm_UpdateSnapshotData(
    PRL_HANDLE hVm,
    PRL_CONST_STR sSnapshotUuid,
    PRL_CONST_STR sNewName,
    PRL_CONST_STR sNewDescription
);
```

File

PrlApiVm.h

Parameters

sSnapshotUuid

The ID of the snapshot to modify. To obtain a list of the available snapshots and their ID's, use the PrlVm_GetSnapshotsTreeEx (p. 468) function.

sNewName

New snapshot name.

sNewDescription

New snapshot description.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible value are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_CreateSnapshot (p. 458)

PrlVm_DeleteSnapshot (p. 461)

PrlVm_GetSnapshotsTreeEx (p. 468)

PrlVm_SwitchToSnapshot (p. 497)

PrlVm_ValidateConfig Function

Validates the specified section of a virtual machine configuration.

Syntax

```
PRL_HANDLE PrlVm_ValidateConfig(
    PRL_HANDLE hVm,
    PRL_VM_CONFIG_SECTIONS nSection
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

nSection

The section in the virtual machine configuration to verify.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

When creating or modifying a virtual machine configuration, this function can be used to verify that a particular section of the configuration is valid. Use this function before sending the virtual machine object containing the configuration information to the Virtuozzo Service. If the configuration is valid, you can commit the changes. If the configuration is not valid, make the necessary modifications and retry.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or the specified configuration section doesn't exist.

PRL_ERR_PARSE_VM_CONFIG (p. 1337) - error parsing configuration data.

PRL_ERR_VMCONF_VALIDATION_FAILED (p. 1476) - invalid configuration; the description of the problem can be retrieved from from the PHT_JOB (p. 138) object (see below).

PRL_ERR_SUCCESS (p. 1365) - specified configuration is valid.

In the event of validation failure, the PHT_JOB (p. 138) object will contain the description of the problem. To get the description:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle to the PHT_RESULT (p. 210) object.
2. Use the PrlResult_GetParamsCount (p. 216) function to obtain the result count.
3. Use the PrlResult_GetParamByIndex (p. 213) to obtain a handle of type PHT_EVENT (p. 72) (if there are multiple results, call the function for each one).
4. Use the PrlEvent_GetErrString (p. 81) function to get the validation error description.

PrlVmCfg_IsHighResolutionEnabled Function

Determines disable or enable high resolution video mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsHighResolutionEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

pbEnabled

A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_IsVerticalSynchronizationEnabled Function

Determines disable or enable vertical synchronization video mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsVerticalSynchronizationEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

pbEnabled

A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_SetHighResolutionEnabled Function

Disables or enables high resolution video mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetHighResolutionEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

bEnabled

Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_SetVerticalSynchronizationEnabled Function

Disables or enables vertical synchronization video mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetVerticalSynchronizationEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

bEnabled

Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PHT_VIRTUAL_NETWORK

The PHT_VIRTUAL_NETWORK handle is used to manage virtual networks. A new handle is obtained by calling the PrlVirtNet_Create (p. 508) function. A handle to an existing virtual network is obtained by calling the PrlSrv_GetVirtualNetworkList (p. 263) function.

Functions

	Name	Description
	PrlVirtNet_Create (p. 508)	Creates a new handle of type PHT_VIRTUAL_NETWORK.
	PrlVirtNet_GetAdapterIndex (p. 509)	Returns a numeric index assigned to the network adapter in the specified virtual network.
	PrlVirtNet_GetAdapterName (p. 510)	Returns the name of the network adapter in the specified virtual network.
	PrlVirtNet_GetBoundAdapterInfo (p. 510)	Obtains physical adapter info which bound to the virtual network object.
	PrlVirtNet_GetBoundCardMac (p. 511)	Returns the bound card MAC address of the specified virtual network.
	PrlVirtNet_GetDescription (p. 512)	Returns the description of the specified virtual network.
	PrlVirtNet_GetDhcpIP6Address (p. 512)	Returns the DCHP IPv6 address of the specified virtual network.
	PrlVirtNet_GetDhcpIPAddress (p. 513)	Returns the DCHP IP address of the specified virtual network.
	PrlVirtNet_GetHostIP6Address (p. 514)	Returns the host IPv6 address of the specified virtual network.
	PrlVirtNet_GetHostIPAddress (p. 514)	Returns the host IP address of the specified virtual network.
	PrlVirtNet_GetIP6NetMask (p. 515)	Returns the IPv6 net mask of the specified virtual network.
	PrlVirtNet_GetIP6ScopeEnd (p. 516)	Returns the DHCP end IPv6 address of the specified virtual network.
	PrlVirtNet_GetIP6ScopeStart (p. 516)	Returns the DHCP start IPv6 address of the specified virtual network.
	PrlVirtNet_GetIPNetMask (p. 517)	Returns the IP net mask of the specified virtual network.
	PrlVirtNet_GetIPScopeEnd (p. 518)	Returns the DHCP end IP address of the specified virtual network.
	PrlVirtNet_GetIPScopeStart (p. 518)	Returns the DHCP start IP address of the specified virtual network.
	PrlVirtNet_GetNetworkId (p. 519)	Returns the ID of the specified virtual network.
	PrlVirtNet_GetNetworkType (p. 520)	Returns the virtual network type.
	PrlVirtNet_GetPortForwardList (p. 520)	Returns the port forward entries list.
	PrlVirtNet_GetVlanTag (p. 521)	Returns the VLAN tag of the specified virtual network.
	PrlVirtNet_IsAdapterEnabled (p. 521)	Determines whether the specified virtual network adapter is enabled or disabled.

	PrlVirtNet_IsDHCP6ServerEnabled (p. 522)	Determines whether the specified virtual network DHCPv6 server is enabled or disabled.
	PrlVirtNet_IsDHCPServerEnabled (p. 522)	Determines whether the specified virtual network DHCP server is enabled or disabled.
	PrlVirtNet_IsEnabled (p. 523)	Determines whether the specified virtual network is enabled or disabled.
	PrlVirtNet_IsNATServerEnabled (p. 524)	Determines whether the specified virtual network NAT server is enabled or disabled.
	PrlVirtNet_SetAdapterEnabled (p. 524)	Allows to enable or disable a specified virtual network adapter.
	PrlVirtNet_SetAdapterIndex (p. 525)	Sets the specified adapter index.
	PrlVirtNet_SetAdapterName (p. 525)	Sets the specified virtual network adapter name.
	PrlVirtNet_SetBoundCardMac (p. 526)	Sets the specified virtual network bound card MAC address.
	PrlVirtNet_SetDescription (p. 526)	Sets the specified virtual network description.
	PrlVirtNet_SetDHCP6ServerEnabled (p. 527)	Allows to enable or disable a specified virtual network DHCPv6 server.
	PrlVirtNet_SetDhcpIP6Address (p. 527)	Sets the specified virtual network DHCP IPv6 address.
	PrlVirtNet_SetDhcpIPAddress (p. 528)	Sets the specified virtual network DHCP IP address.
	PrlVirtNet_SetDHCPServerEnabled (p. 528)	Allows to enable or disable a specified virtual network DHCP server.
	PrlVirtNet_SetEnabled (p. 529)	Allows to enable or disable a specified virtual network.
	PrlVirtNet_SetHostIP6Address (p. 529)	Sets the specified virtual network host IPv6 address.
	PrlVirtNet_SetHostIPAddress (p. 530)	Sets the specified virtual network host IP address.
	PrlVirtNet_SetIP6NetMask (p. 531)	Sets the specified virtual network IPv6 net mask.
	PrlVirtNet_SetIP6ScopeEnd (p. 531)	Sets the specified virtual network DHCP end IPv6 address.
	PrlVirtNet_SetIP6ScopeStart (p. 532)	Sets the specified virtual network DHCP start IPv6 address.
	PrlVirtNet_SetIPNetMask (p. 532)	Sets the specified virtual network IP net mask.
	PrlVirtNet_SetIPScopeEnd (p. 533)	Sets the specified virtual network DHCP end IP address.
	PrlVirtNet_SetIPScopeStart (p. 533)	Sets the specified virtual network DHCP start IP address.
	PrlVirtNet_SetNATServerEnabled (p. 534)	Allows to enable or disable a specified virtual network NAT server.
	PrlVirtNet_SetNetworkId (p. 534)	Sets the specified virtual network ID.
	PrlVirtNet_SetNetworkType (p. 535)	Sets the specified virtual network type.
	PrlVirtNet_SetPortForwardList (p. 535)	Sets the port forward entries list.
	PrlVirtNet_SetVlanTag (p. 536)	Sets the VLAN tag for the specified virtual network.

PrlVirtNet_Create Function

Creates a new handle of type PHT_VIRTUAL_NETWORK (p. 507).

Syntax

```
PRL_RESULT PrlVirtNet_Create(
```

```
    PRL_HANDLE_PTR phVirtNet
);
```

File

PrlApiNet.h

Parameters

phVirtNet

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to create and configure a virtual network. If you would like to create a new virtual network, use this function to create the handle first and then use the handle functions to configure the network. If you would like to modify the configuration of an existing virtual network, use the PrlSrv_GetVirtualNetworkList (p. 263) function to obtain a handle to the virtual network of interest.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

PrlVirtNet_GetAdapterIndex Function

Returns a numeric index assigned to the network adapter in the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetAdapterIndex(
    PRL_HANDLE hVirtNet,
    PRL_UINT32_PTR pnAdapterIndex
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pnAdapterIndex

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Special values for this parameter in PRL_ADAPTER_INDEX (p. 1036) enum.

PrlVirtNet_GetAdapterName Function

Returns the name of the network adapter in the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetAdapterName(
    PRL_HANDLE hVirtNet,
    PRL_STR sAdapterName,
    PRL_UINT32_PTR pnAdapterNameBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sAdapterName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnAdapterNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetBoundAdapterInfo Function

Obtains physical adapter info which bound to the virtual network object.

Syntax

```
PRL_RESULT PrlVirtNet_GetBoundAdapterInfo(
    PRL_HANDLE hVirtNet,
    PRL_HANDLE hSrvConfig,
    PRL_HANDLE_PTR phNetAdapter
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314) which provides host hardware information.

phNetAdapter

[out] A pointer to a variable that receives the new handle of PHT_HW_NET_ADAPTER (p. 126) object.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network object or invalid handle of host hardware info or null pointer was passed.

PRL_ERR_NETWORK_ADAPTER_NOT_FOUND (p. 1323) - correspond network adapter is absent at provided host

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the object of network adapter.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

This method provided for convenience in order do not create own algorithm of bound to the virtual network physical adapter information.

PrlVirtNet_GetBoundCardMac Function

Returns the bound card MAC address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetBoundCardMac(
    PRL_HANDLE hVirtNet,
    PRL_STR sBoundCardMac,
    PRL_UINT32_PTR pnBoundCardMacBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sBoundCardMac

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnBoundCardMacBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetDescription Function

Returns the description of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetDescription(
    PRL_HANDLE hVirtNet,
    PRL_STR sDescription,
    PRL_UINT32_PTR pnDescriptionBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sDescription

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDescriptionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetDhcpIP6Address Function

Returns the DCHP IPv6 address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetDhcpIP6Address(
    PRL_HANDLE hVirtNet,
    PRL_STR sDhcpIPAddress,
    PRL_UINT32_PTR pnDhcpIPAddressBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sDhcpIPAddress

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDhcpIPAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetDhcpIPAddress Function

Returns the DCHP IP address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetDhcpIPAddress(
    PRL_HANDLE hVirtNet,
    PRL_STR sDhcpIPAddress,
    PRL_UINT32_PTR pnDhcpIPAddressBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sDhcpIPAddress

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDhcpIPAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetHostIP6Address Function

Returns the host IPv6 address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetHostIP6Address(
    PRL_HANDLE hVirtNet,
    PRL_STR sHostIPAddress,
    PRL_UINT32_PTR pnHostIPAddressBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sHostIPAddress

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnHostIPAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetHostIPAddress Function

Returns the host IP address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetHostIPAddress(
    PRL_HANDLE hVirtNet,
    PRL_STR sHostIPAddress,
    PRL_UINT32_PTR pnHostIPAddressBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sHostIPAddress

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnHostIPAddressBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetIP6NetMask Function

Returns the IPv6 net mask of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetIP6NetMask(
    PRL_HANDLE hVirtNet,
    PRL_STR sIPNetMask,
    PRL_UINT32_PTR pnIPNetMaskBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPNetMask

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIPNetMaskBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetIP6ScopeEnd Function

Returns the DHCP end IPv6 address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetIP6ScopeEnd(
    PRL_HANDLE hVirtNet,
    PRL_STR sIPScopeEnd,
    PRL_UINT32_PTR pnIPScopeEndBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeEnd

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIPScopeEndBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetIP6ScopeStart Function

Returns the DHCP start IPv6 address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetIP6ScopeStart(
    PRL_HANDLE hVirtNet,
    PRL_STR sIPScopeStart,
    PRL_UINT32_PTR pnIPScopeStartBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeStart

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIPScopeStartBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetIPNetMask Function

Returns the IP net mask of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetIPNetMask(
    PRL_HANDLE hVirtNet,
    PRL_STR sIPNetMask,
    PRL_UINT32_PTR pnIPNetMaskBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPNetMask

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIPNetMaskBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetIPScopeEnd Function

Returns the DHCP end IP address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetIPScopeEnd(
    PRL_HANDLE hVirtNet,
    PRL_STR sIPScopeEnd,
    PRL_UINT32_PTR pnIPScopeEndBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeEnd

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIPScopeEndBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetIPScopeStart Function

Returns the DHCP start IP address of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetIPScopeStart(
    PRL_HANDLE hVirtNet,
    PRL_STR sIPScopeStart,
    PRL_UINT32_PTR pnIPScopeStartBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeStart

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnIPScopeStartBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetNetworkId Function

Returns the ID of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetNetworkId(
    PRL_HANDLE hVirtNet,
    PRL_STR sNetworkId,
    PRL_UINT32_PTR pnNetworkIdBufLength
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sNetworkId

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnNetworkIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetNetworkType Function

Returns the virtual network type.

Syntax

```
PRL_RESULT PrlVirtNet_GetNetworkType(
    PRL_HANDLE hVirtNet,
    PRL_NET_VIRTUAL_NETWORK_TYPE_PTR pnNetworkType
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pnNetworkType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Network types are defined in the PRL_NET_VIRTUAL_NETWORK_TYPE (p. 1081) enumeration. The following describes network types are used:

- PVN_BRIDGED_ETHERNET
- PVN_HOST_ONLY

PrlVirtNet_GetPortForwardList Function

Returns the port forward entries list.

Syntax

```
PRL_RESULT PrlVirtNet_GetPortForwardList(
    PRL_HANDLE hVirtNet,
    PRL_PORT_FORWARDING_TYPE nPortFwdType,
    PRL_HANDLE_PTR phPortFwdList
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

nPortFwdType

A port forward type (PRL_PORT_FORWARDING_TYPE (p. 1083) enumeration).

phPortFwdList

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list or a port forward handle object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_GetVlanTag Function

Returns the VLAN tag of the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_GetVlanTag(
    PRL_HANDLE hVirtNet,
    PRL_UINT16_PTR pnVlanTag
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pnVlanTag

[out] The buffer for storing result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Since the same physical adapter can be used at several virtual networks pair can identify adapter.

PrlVirtNet_IsAdapterEnabled Function

Determines whether the specified virtual network adapter is enabled or disabled.

Syntax

```
PRL_RESULT PrlVirtNet_IsAdapterEnabled(
    PRL_HANDLE hVirtNet,
```

```
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the adapter is enabled. PRL_FALSE (p. 1626) indicates that the adapter is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_IsDHCP6ServerEnabled Function

Determines whether the specified virtual network DHCPv6 server is enabled or disabled.

Syntax

```
PRL_RESULT PrlVirtNet_IsDHCP6ServerEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the DHCP server is enabled. PRL_FALSE (p. 1626) indicates that the DHCP server is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_IsDHCPServerEnabled Function

Determines whether the specified virtual network DHCP server is enabled or disabled.

Syntax

```
PRL_RESULT PrlVirtNet_IsDHCPServerEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the DHCP server is enabled. PRL_FALSE (p. 1626) indicates that the DHCP server is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_IsEnabled Function

Determines whether the specified virtual network is enabled or disabled.

Syntax

```
PRL_RESULT PrlVirtNet_IsEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the virtual network is enabled. PRL_FALSE (p. 1626) indicates that the virtual network is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_IsNATServerEnabled Function

Determines whether the specified virtual network NAT server is enabled or disabled.

Syntax

```
PRL_RESULT PrlVirtNet_IsNATServerEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the NAT server is enabled. PRL_FALSE (p. 1626) indicates that the NAT server is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetAdapterEnabled Function

Allows to enable or disable a specified virtual network adapter.

Syntax

```
PRL_RESULT PrlVirtNet_SetAdapterEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL bEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

bEnabled

A boolean value indicating the type of action to perform. PRL_TRUE (p. 1652) will enable the adapter. PRL_FALSE (p. 1626) will disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetAdapterIndex Function

Sets the specified adapter index.

Syntax

```
PRL_RESULT PrlVirtNet_SetAdapterIndex(
    PRL_HANDLE hVirtNet,
    PRL_UINT32 nAdapterIndex
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

nAdapterIndex

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Special values for this parameter in PRL_ADAPTER_INDEX (p. 1036) enum.

PrlVirtNet_SetAdapterName Function

Sets the specified virtual network adapter name.

Syntax

```
PRL_RESULT PrlVirtNet_SetAdapterName(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sAdapterName
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sAdapterName

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetBoundCardMac Function

Sets the specified virtual network bound card MAC address.

Syntax

```
PRL_RESULT PrlVirtNet_SetBoundCardMac(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sBoundCardMac
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sBoundCardMac

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

MAC address gets by PrlSrvCfgNet_GetMacAddress (p. 128) functions for given network adapter.

PrlVirtNet_SetDescription Function

Sets the specified virtual network description.

Syntax

```
PRL_RESULT PrlVirtNet_SetDescription(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sDescription
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sDescription

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetDHCP6ServerEnabled Function

Allows to enable or disable a specified virtual network DHCPv6 server.

Syntax

```
PRL_RESULT PrlVirtNet_SetDHCP6ServerEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL bEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virual network.

bEnabled

A boolean value indicating the type of action to perform. PRL_TRUE (p. 1652) will enable the DHCP server. PRL_FALSE (p. 1626) will disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetDhcpIP6Address Function

Sets the specified virtual network DHCP IPv6 address.

Syntax

```
PRL_RESULT PrlVirtNet_SetDhcpIP6Address(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sDhcpIPAddress
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sDhcpIPAddress

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetDhcpIPAddress Function

Sets the specified virtual network DHCP IP address.

Syntax

```
PRL_RESULT PrlVirtNet_SetDhcpIPAddress(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sDhcpIPAddress
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sDhcpIPAddress

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetDHCPServerEnabled Function

Allows to enable or disable a specified virtual network DHCP server.

Syntax

```
PRL_RESULT PrlVirtNet_SetDHCPServerEnabled(
    PRL_HANDLE hVirtNet,
```

```
    PRL_BOOL bEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virual network.

bEnabled

A boolean value indicating the type of action to perform. PRL_TRUE (p. 1652) will enable the DHCP server. PRL_FALSE (p. 1626) will disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetEnabled Function

Allows to enable or disable a specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_SetEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL bEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

bEnabled

A boolean value indicating the type of action to perform. PRL_TRUE (p. 1652) will enable the virtual network. PRL_FALSE (p. 1626) will disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetHostIP6Address Function

Sets the specified virtual network host IPv6 address.

Syntax

```
PRL_RESULT PrlVirtNet_SetHostIP6Address(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sHostIPAddress
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sHostIPAddress

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetHostIPAddress Function

Sets the specified virtual network host IP address.

Syntax

```
PRL_RESULT PrlVirtNet_SetHostIPAddress(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sHostIPAddress
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sHostIPAddress

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetIP6NetMask Function

Sets the specified virtual network IPv6 net mask.

Syntax

```
PRL_RESULT PrlVirtNet_SetIP6NetMask(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sIPNetMask
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPNetMask

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetIP6ScopeEnd Function

Sets the specified virtual network DHCP end IPv6 address.

Syntax

```
PRL_RESULT PrlVirtNet_SetIP6ScopeEnd(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sIPScopeEnd
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeEnd

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetIP6ScopeStart Function

Sets the specified virtual network DHCP start IPv6 address.

Syntax

```
PRL_RESULT PrlVirtNet_SetIP6ScopeStart(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sIPScopeStart
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeStart

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetIPNetMask Function

Sets the specified virtual network IP net mask.

Syntax

```
PRL_RESULT PrlVirtNet_SetIPNetMask(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sIPNetMask
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPNetMask

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetIPScopeEnd Function

Sets the specified virtual network DHCP end IP address.

Syntax

```
PRL_RESULT PrlVirtNet_SetIPScopeEnd(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sIPScopeEnd
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeEnd

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetIPScopeStart Function

Sets the specified virtual network DHCP start IP address.

Syntax

```
PRL_RESULT PrlVirtNet_SetIPScopeStart(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sIPScopeStart
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sIPScopeStart

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetNATServerEnabled Function

Allows to enable or disable a specified virtual network NAT server.

Syntax

```
PRL_RESULT PrlVirtNet_SetNATServerEnabled(
    PRL_HANDLE hVirtNet,
    PRL_BOOL bEnabled
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virual network.

bEnabled

A boolean value indicating the type of action to perform. PRL_TRUE (p. 1652) will enable the NAT server. PRL_FALSE (p. 1626) will disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetNetworkId Function

Sets the specified virtual network ID.

Syntax

```
PRL_RESULT PrlVirtNet_SetNetworkId(
    PRL_HANDLE hVirtNet,
    PRL_CONST_STR sNetworkId
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

sNetworkId

The value to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetNetworkType Function

Sets the specified virtual network type.

Syntax

```
PRL_RESULT PrlVirtNet_SetNetworkType(
    PRL_HANDLE hVirtNet,
    PRL_NET_VIRTUAL_NETWORK_TYPE nNetworkType
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

nNetworkType

The value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Network types are defined in the PRL_NET_VIRTUAL_NETWORK_TYPE (p. 1081) enumeration. The following describes network types are used:

- PVN_BRIDGED_ETHERNET
- PVN_HOST_ONLY

PrlVirtNet_SetPortForwardList Function

Sets the port forward entries list.

Syntax

```
PRL_RESULT PrlVirtNet_SetPortForwardList(
    PRL_HANDLE hVirtNet,
    PRL_PORT_FORWARDING_TYPE nPortFwdType,
```

```
    PRL_HANDLE hPortFwdList
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

nPortFwdType

A port forward type (PRL_PORT_FORWARDING_TYPE (p. 1083) enumeration).

hPortFwdList

The list of PHT_PORT_FORWARDING (p. 185) handles.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a port forward entry object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVirtNet_SetVlanTag Function

Sets the VLAN tag for the specified virtual network.

Syntax

```
PRL_RESULT PrlVirtNet_SetVlanTag(
    PRL_HANDLE hVirtNet,
    PRL_UINT16 nVlanTag
);
```

File

PrlApiNet.h

Parameters

hVirtNet

A handle of type PHT_VIRTUAL_NETWORK (p. 507) identifying the virtual network.

nVlanTag

[in] Setting value.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Since the same physical adapter can be used at several virtual networks pair can identify adapter.

PHT_VM_CONFIGURATION

The PHT_VM_CONFIGURATION handle is used to manage the virtual machine configuration. The handle is obtained by calling the PrlVm_GetConfig (p. 464) function.

Functions

	Name	Description
	PrlVmCfg_AddDefaultDevice (p. 547)	The PrlVmCfg_AddDefaultDevice function automates the task of setting devices in a virtual machine.
	PrlVmCfg_AddDefaultDeviceEx (p. 548)	The PrlVmCfg_AddDefaultDeviceEx function automates the task of setting devices in a virtual machine.
	PrlVmCfg_ApplyConfigSample (p. 549)	Specify the Configuration sample file
	PrlVmCfg_CreateBootDev (p. 550)	Creates a new handle of type PHT_BOOT_DEVICE (p. 14) and adds it to the specified virtual machine boot device list.
	PrlVmCfg_CreateShare (p. 550)	Creates a new handle of type PHT_SHARE (p. 350) and adds it to the specified virtual machine list of shares.
	PrlVmCfg_CreateVmDev (p. 551)	Creates a new virtual device handle of the specified type.
	PrlVmCfg_Get3DAccelerationMode (p. 552)	Gets 3D graphical acceleration technology mode for the specified virtual machine.
	PrlVmCfg_GetAccessRights (p. 552)	Obtains a handle of type PHT_ACCESS_RIGHTS (p. 9).
	PrlVmCfg_GetActionOnGuestCrash (p. 553)	%VM_ONLY% Determines the on-crash action for the specified virtual machine
	PrlVmCfg_GetActionOnStopMode (p. 553)	Gets action on stop mode.
	PrlVmCfg_GetActionOnWindowClose (p. 554)	Determines the action on Virtuozzo Application window close for the specified virtual machine.
	PrlVmCfg_GetAllDevices (p. 554)	Obtains a handles list of all virtual devices in a virtual machine.
	PrlVmCfg_GetAppTemplateList (p. 555)	Returns the list of application templates installed for the specified Container.
	PrlVmCfg_GetAutoCompressInterval (p. 555)	Determines the interval at which compacting virtual disks are doing by Automatic HDD compress.
	PrlVmCfg_GetAutoStart (p. 556)	Determines if the specified virtual machine is set to start automatically on Virtuozzo Service startup.
	PrlVmCfg_GetAutoStartDelay (p. 557)	Returns the time delay used during the virtual machine automatic startup.
	PrlVmCfg_GetAutoStop (p. 557)	Determines the mode of the automatic shutdown for the specified virtual machine.
	PrlVmCfg_GetBackgroundPriority (p. 558)	Determines the specified virtual machine background process priority type.
	PrlVmCfg_GetBootDev (p. 558)	Obtains a handle of type PHT_BOOT_DEVICE (p. 14) containing information about a specified device from the virtual machine boot device priority list.
	PrlVmCfg_GetBootDevCount (p. 559)	Determines the number of devices in the virtual machine boot device priority list.

	PrlVmCfg_GetCapabilitiesMask (p. 559)	Returns a capabilities mask specified inside the CT.
	PrlVmCfg_GetConfigValidity (p. 560)	The PrlVmCfg_GetConfigValidity function returns an error code.
	PrlVmCfg_GetConfirmationsList (p. 561)	Obtains a handles list of operations with virtual machine which required administrator confirmation
	PrlVmCfg_GetCpuAccelLevel (p. 561)	Determines the virtual machine CPU acceleration level.
	PrlVmCfg_GetCpuCount (p. 562)	Determines the number of CPUs in the virtual machine.
	PrlVmCfg_GetCpuLimit (p. 562)	Determines the CPU usage limit of a virtual machine, in percent.
	PrlVmCfg_GetCpuLimitEx (p. 563)	Determines the specified virtual machine CPU limit.
	PrlVmCfg_GetCpuMask (p. 563)	Returns the specified virtual machine CPU affinity mask
	PrlVmCfg_GetCpuMode (p. 564)	Determines the specified virtual machine CPU mode (32 bit or 64 bit).
	PrlVmCfg_GetCpuUnits (p. 565)	Determines the number of CPU units allocated to a virtual machine.
	PrlVmCfg_GetCtId (p. 565)	Returns the Container id of the specified virtual machine.
	PrlVmCfg_GetCustomProperty (p. 566)	Returns the virtual machine custom property information.
	PrlVmCfg_GetDefaultHddSize (p. 567)	The PrlVmCfg_GetDefaultHddSize function returns the default hard disk size for to the specified OS type and version.
	PrlVmCfg_GetDefaultMemSize (p. 567)	The PrlVmCfg_GetDefaultMemSize function returns the default RAM size for to the specified OS type and version.
	PrlVmCfg_GetDefaultVideoRamSize (p. 568)	The PrlVmCfg_GetDefaultVideoRamSize function returns the default video RAM size for the specified OS type and version.
	PrlVmCfg_GetDescription (p. 569)	Returns the specified virtual machine description.
	PrlVmCfg_GetDevByType (p. 570)	Obtains a handle to a virtual device in a virtual machine according to the specified device type and index.
	PrlVmCfg_GetDevsCount (p. 570)	Determines the total number of devices of any type installed in the virtual machine.
	PrlVmCfg_GetDevsCountByType (p. 571)	Obtains the number of virtual devices of the specified type available in the specified virtual machine.
	PrlVmCfg_GetDisplayDev (p. 572)	Obtains a handle of type PHT_VIRTUAL_DEV_DISPLAY containing information about a display device in a virtual machine.
	PrlVmCfg_GetDisplayDevsCount (p. 572)	Determines the number of display devices in a virtual machine.
	PrlVmCfg_GetDnsServers (p. 573)	Obtains the list of of globally defined DNS servers which is assigned to a Virtuozzo Container.
	PrlVmCfg_GetExternalBootDevice (p. 573)	Returns an external boot device system name.
	PrlVmCfg_GetFeaturesMask (p. 574)	Returns a features mask specified inside the CT.
	PrlVmCfg_GetFloppyDisk (p. 575)	Obtains a handle of type PHT_VIRTUAL_DEV_FLOPPY (p. 381) containing information about the specified floppy disk drive in a virtual machine.
	PrlVmCfg_GetFloppyDisksCount (p. 575)	Determines the number of floppy disk drives in a virtual machine.

	PrlVmCfg_GetForegroundPriority (p. 576)	Returns foreground processes priority for the specified virtual machine.
	PrlVmCfg_GetFreeDiskSpaceRatio (p. 576)	Determines the free disk space ratio at which compacting virtual disks are doing by Automatic HDD compress.
	PrlVmCfg_GetGenericPciDev (p. 577)	Obtains a handle of type PHT_VIRTUAL_DEV_GENERIC_PCI containing information about a generic PCI device in a virtual machine.
	PrlVmCfg_GetGenericPciDevsCount (p. 578)	Determines the number of generic PCI devices in a virtual machine.
	PrlVmCfg_GetGenericScsiDev (p. 578)	Obtains a handle of type PHT_VIRTUAL_DEV_GENERIC_SCSI containing information about a generic SCSI device in a virtual machine.
	PrlVmCfg_GetGenericScsiDevsCount (p. 579)	Determines the number of generic SCSI devices in a virtual machine.
	PrlVmCfg_GetGuestCpuLimitType (p. 580)	Determines the specified virtual machine CPU limit type.
	PrlVmCfg_GetHardDisk (p. 580)	Obtains a handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) containing the specified virtual hard disk information.
	PrlVmCfg_GetHardDisksCount (p. 581)	Determines the number of virtual hard disks in a virtual machine.
	PrlVmCfg_GetHighAvailabilityPriority (p. 581)	Determines the priority of the virtual machine in the High Availability Cluster.
	PrlVmCfg_GetHomePath (p. 582)	Returns the virtual machine home directory name and path.
	PrlVmCfg_GetHostname (p. 582)	Obtains the hostname of the specified virtual machine.
	PrlVmCfg_GetIcon (p. 583)	Returns the name of the icon file used by the specified virtual machine.
	PrlVmCfg_GetIoLimit (p. 584)	Determines the specified virtual machine I/O limit.
	PrlVmCfg_GetIoPriority (p. 584)	Determines the specified virtual machine I/O priority.
	PrlVmCfg_GetIopsLimit (p. 585)	Determines the specified virtual machine IOPS limit.
	PrlVmCfg_GetLastModifiedDate (p. 585)	Returns the date and time when the specified virtual machine was last modified.
	PrlVmCfg_GetLastModifierName (p. 586)	Returns the name of the user who last modified the specified virtual machine.
	PrlVmCfg_GetLinkedVmUuid (p. 587)	Returns the UUID (universally unique ID) of the parent virtual machine which was used to create current linked cloned virtual machine (if last one is linked clone of some virtual machine - this value will be empty otherwise).
	PrlVmCfg_GetLocation (p. 587)	Returns the virtual machine files location type.
	PrlVmCfg_GetMemGuaranteeSize (p. 588)	Returns the minimum memory limit which is guaranteed for the virtual machine, in percent of guest RAM.
	PrlVmCfg_GetName (p. 589)	Returns the name of the specified virtual machine.
	PrlVmCfg_GetNetAdapter (p. 589)	Obtains a handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) containing information about a virtual network adapter in a virtual machine.
	PrlVmCfg_GetNetAdaptersCount (p. 590)	Determines the number of network adapters in a virtual machine.
	PrlVmCfg_GetNetfilterMode (p. 590)	Returns netfilter mode specified inside the CT.
	PrlVmCfg_GetNetworkRateList (p. 591)	Obtains the list of shaping rated

	PrlVmCfg_GetNodeMask (p. 592)	Returns the specified virtual machine node mask
	PrlVmCfg_GetOfflineServices (p. 592)	Obtains the list of services available in the virtual machine offline management.
	PrlVmCfg_GetOpticalDisk (p. 593)	Obtains a handle of type PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425) containing information a virtual optical disk in a virtual machine.
	PrlVmCfg_GetOpticalDisksCount (p. 594)	Determines the number of optical disks in the specified virtual machine.
	PrlVmCfg_GetOptimizeModifiersMode (p. 594)	Gets optimize modifiers mode.
	PrlVmCfg_GetOptionsOnGuestCrash (p. 595)	%VM_ONLY% Determines the on-crash options for the specified virtual machine
	PrlVmCfg_GetOsTemplate (p. 595)	Returns Operation System template name the Virtuozzo Virtuozzo Container were created on.
	PrlVmCfg_GetOsType (p. 596)	Returns the type of the operating system that the specified virtual machine is running.
	PrlVmCfg_GetOsVersion (p. 596)	Returns the version of the operating system that the specified virtual machine is running.
	PrlVmCfg_GetParallelPort (p. 597)	Obtains a handle of type PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425) containing information about a virtual printer port in the specified virtual machine.
	PrlVmCfg_GetParallelPortsCount (p. 598)	Determines the number of virtual printer ports in the specified virtual machine.
	PrlVmCfg_GetProfile (p. 598)	Get current VM profile.
	PrlVmCfg_GetRamSize (p. 599)	Returns the virtual machine memory (RAM) size, in megabytes.
	PrlVmCfg_GetResource (p. 599)	Returns beancounters barrier and limit for specified resource id.
	PrlVmCfg_GetSearchDomains (p. 600)	Obtains the list of globally defined search domains that will be assigned to a virtual machine on the guest operating system side.
	PrlVmCfg_GetSerialPort (p. 600)	Obtains a handle of type PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425) containing information about a serial port in a virtual machine.
	PrlVmCfg_GetSerialPortsCount (p. 601)	Determines the number of serial ports in a virtual machine.
	PrlVmCfg_GetServerHost (p. 602)	Returns the hostname of the machine hosting the specified virtual machine.
	PrlVmCfg_GetServerUuid (p. 602)	Returns the UUID of the machine hosting the specified virtual machine.
	PrlVmCfg_GetShare (p. 603)	Obtains a handle of type PHT_SHARE (p. 350) containing information about a shared folder.
	PrlVmCfg_GetSharesCount (p. 604)	Determines the number of shared folders in a virtual machine.
	PrlVmCfg_GetSmartGuardInterval (p. 604)	Determines the interval at which snapshots are taken by SmartGuard.
	PrlVmCfg_GetSmartGuardMaxSnapshotsCount (p. 605)	Determines the maximum snapshot count, a SmartGuard setting.

	PrlVmCfg_GetSoundDev (p. 606)	Obtains a handle of type PHT_VIRTUAL_DEV_SOUND (p. 427) containing information about a sound device in a virtual machine.
	PrlVmCfg_GetSoundDevsCount (p. 606)	Determines the number of sound devices in a virtual machine.
	PrlVmCfg_GetStartLoginMode (p. 607)	Returns the automatic startup login mode for the specified virtual machine.
	PrlVmCfg_GetStartUserLogin (p. 607)	Returns the user name used during the specified virtual machine automatic startup.
	PrlVmCfg_GetSystemFlags (p. 608)	Returns the virtual machine system flags.
	PrlVmCfg_GetTimeSyncInterval (p. 609)	Obtains the time synchronization interval between the host and a guest OS.
	PrlVmCfg_GetUndoDisksMode (p. 609)	Determines the current undo-disks mode for the specified virtual machine.
	PrlVmCfg_GetUptime (p. 610)	Returns the virtual machine uptime in seconds since some datetime (see PrlVmCfg_GetUptimeStartDate (p. 610)).
	PrlVmCfg_GetUptimeStartDate (p. 610)	Returns the date and time when uptime counter calculation was started for the specified virtual machine.
	PrlVmCfg_GetUsbDevice (p. 611)	Obtains a handle of type PHT_VIRTUAL_DEV_USB_DEVICE (p. 430) containing information about a USB device in a virtual machine.
	PrlVmCfg_GetUsbDevicesCount (p. 612)	Determines the number of USB devices in a virtual machine.
	PrlVmCfg_GetUuid (p. 612)	Returns the UUID (universally unique ID) of the specified virtual machine.
	PrlVmCfg_GetVideoRamSize (p. 613)	Returns the video memory size of the specified virtual machine.
	PrlVmCfg_GetVmInfo (p. 613)	Obtains a handle of type PHT_VM_INFO (p. 719) containing the specified virtual machine state and access rights information.
	PrlVmCfg_GetVmType (p. 614)	Get the Virtual Machine type.
	PrlVmCfg_GetVNCHostName (p. 614)	Returns the VNC hostname of the specified virtual machine.
	PrlVmCfg_GetVNCMode (p. 615)	Returns the VNC mode of the specified virtual machine.
	PrlVmCfg_GetVNCPassword (p. 616)	Returns the VNC password for the specified virtual machine.
	PrlVmCfg_GetVNCPort (p. 616)	Returns the VNC port number for the specified virtual machine.
	PrlVmCfg_GetVNCWebSocketPort (p. 617)	Returns the VNC WebSocket port number for the specified virtual machine.
	PrlVmCfg_GetWindowMode (p. 617)	Returns the current window mode the specified virtual machine is in.
	PrlVmCfg_IsAdaptiveHypervisorEnabled (p. 618)	Determines disable or enable adaptive hypervisor.
	PrlVmCfg_IsAllowSelectBootDevice (p. 618)	Determines whether select boot device should be shown on virtual machine startup.
	PrlVmCfg_IsAutoApplyIpOnly (p. 619)	Determines if the Guest configured to automatically apply IP addrs network settings inside guest.
	PrlVmCfg_IsAutoCompressEnabled (p. 620)	Determines whether the Automatic HDD compress feature is enabled or not in a virtual machine.
	PrlVmCfg_IsBatteryStatusEnabled (p. 620)	Determines disable or enable battery status.
	PrlVmCfg_IsCloseAppOnShutdown (p. 621)	Determines whether the Virtuozzo application (such as Virtuozzo Desktop) will be automatically closed on the virtual machine shutdown.

	PrlVmCfg_IsConfigInvalid (p. 621)	The PrlVmCfg_IsConfigInvalid function is used to check the • error code of PrlVmCfg_GetConfigValidity (p. 560) function that • indicates the invalid status of the config: • config doesn't exist, parsing error, invalid server uuid, • invalid VM uuid. • Parameters • nErrCode : The nErrCode value from PrlVmCfg_GetConfigValidity (p. 560)() • pbInvalid : [out] validity sign.
	PrlVmCfg_IsCpuHotplugEnabled (p. 622)	Determines whether the CPU hotplug is available in the virtual machine.
	PrlVmCfg_IsCpuVtxEnabled (p. 622)	Determines whether the x86 virtualization (such as Vt-x) is available in the virtual machine CPU.
	PrlVmCfg_IsCustomProfile (p. 623)	Determines were settings modified after profile was applied or not.
	PrlVmCfg_IsDefaultDeviceNeeded (p. 623)	Determines whether the virtual machine requires a default device of the specified type if it's going to run an operating system of the specified type.
	PrlVmCfg_IsDisableAPIC (p. 624)	Determines whether the APIC is enabled or disabled during the virtual machine runtime.
	PrlVmCfg_IsDisableSpeaker (p. 625)	Determines whether the speaker is enabled or disabled during the virtual machine runtime.
	PrlVmCfg_IsDiskCacheWriteBack (p. 625)	Determines if disk cache write-back is enabled in the specified virtual machine.
	PrlVmCfg_IsEfiEnabled (p. 626)	Determines whether EFI boot is switched on.
	PrlVmCfg_IsGuestSharingAutoMount (p. 626)	Determines if host shared folders are mounted automatically in a given virtual machine.
	PrlVmCfg_IsGuestSharingEnabled (p. 627)	Determines if guest sharing is enabled (the guest OS disk drives are visible in the host OS).
	PrlVmCfg_IsGuestSharingEnableSpotlight (p. 628)	Determines if the virtual disks in a virtual machine will be added to Spotlight search subsystem (Mac OS X feature).
	PrlVmCfg_IsHighAvailabilityEnabled (p. 628)	Determines whether the High Availability feature is enabled or not for a virtual machine.
	PrlVmCfg_IsHostSharingEnabled (p. 629)	Determines if host sharing is enabled (host shared folders are visible in the guest OS).
	PrlVmCfg_IsIsolatedVmEnabled (p. 630)	Determines disable or enable isolated VM.
	PrlVmCfg_IsLockGuestOnSuspendEnabled (p. 630)	Determines disable or enable lock guest on suspend.
	PrlVmCfg_IsLockInFullScreenMode (p. 631)	Determines whether the lock in screen mode sign is set or not in a virtual machine configuration.
	PrlVmCfg_IsLongerBatteryLifeEnabled (p. 631)	Determines disable or enable longer battery life.
	PrlVmCfg_IsMapSharedFoldersOnLetters (p. 632)	Determines whether host disks shared with the guest Windows OS will be mapped to drive letters.
	PrlVmCfg_IsNestedVirtualizationEnabled (p. 632)	Determines disable or enable nested virtualization.
	PrlVmCfg_IsOfflineManagementEnabled (p. 633)	Determines whether the offline management feature is enabled for a virtual machine.

	PrlVmCfg_IsOsResInFullScrMode (p. 634)	Determines wether the virtual machine OS resolution is in full screen mode.
	PrlVmCfg_IsPMUVirtualizationEnabled (p. 634)	Determines disable or enable PMU virtualization.
	PrlVmCfg_IsRamHotplugEnabled (p. 635)	Determines whether the memory (RAM) hotplug is available in the virtual machine.
	PrlVmCfg_IsRateBound (p. 635)	Determines whether the ratebound is enabled or not.
	PrlVmCfg_IsScrResEnabled (p. 636)	Determines if additional screen resolution support is enabled in a virtual machine.
	PrlVmCfg_IsShareAllHostDisks (p. 636)	Determines whether all host disks will be present at guest OS as shares.
	PrlVmCfg_IsShareClipboard (p. 637)	Determines whether the clipboard sharing feature is enabled or disabled in a virtual machine.
	PrlVmCfg_IsSharedProfileEnabled (p. 637)	Determines whether the Shared Profile feature is enabled in a virtual machine.
	PrlVmCfg_IsShareUserHomeDir (p. 638)	Determines whether the host user home directory will be available in the guest OS as a share.
	PrlVmCfg_IsSmartGuardEnabled (p. 639)	Determines whether the SmartGuard feature is enabled or not in a virtual machine.
	PrlVmCfg_IsSmartGuardNotifyBeforeCreation (p. 639)	Determines whether the user will be notified on automatic snapshot creation, a SmartGaurd setting.
	PrlVmCfg_IsSmartMountDVDsEnabled (p. 640)	Determines whether the SmartMount CD/DVD drives is enabled or not.
	PrlVmCfg_IsSmartMountEnabled (p. 641)	Determines whether the SmartMount feature is enabled or not.
	PrlVmCfg_IsSmartMountNetworkSharesEnabled (p. 641)	Determines whether the SmartMount network shares is enabled or not.
	PrlVmCfg_IsSmartMountRemovableDrivesEnabled (p. 642)	Determines whether the SmartMount removable drives is enabled or not.
	PrlVmCfg_IsStartInDetachedWindowEnabled (p. 642)	Determines disable or enable start in detached window.
	PrlVmCfg_IsSwitchOffWindowsLogoEnabled (p. 643)	Determines disable or enable Windows logo.
	PrlVmCfg_IsTemplate (p. 643)	Determines if the specified virtual machine object contains a virtual machine or a template configuration information.
	PrlVmCfg_IsTimeSynchronizationEnabled (p. 644)	Determines whether the time synchronization feature is enabled in a virtual machine.
	PrlVmCfg_IsTimeSyncSmartModeEnabled (p. 644)	Determines whether the smart time synchronization is enabled in a virtual machine.
	PrlVmCfg_IsToolsAutoUpdateEnabled (p. 645)	Enables or disables the Virtuozzo Tools AutoUpdate feature for a virtual machine.
	PrlVmCfg_IsUseDefaultAnswers (p. 646)	Determines whether the use default answers mechanism active for the virtual machine.
	PrlVmCfg_IsUseDesktop (p. 646)	Determines whether the "use desktop in shared profile" feature is enabled or not.
	PrlVmCfg_IsUseDocuments (p. 647)	Determines whether "use documents in shared profile" feature is enabled or not.
	PrlVmCfg_IsUseDownloads (p. 647)	Determines whether the "used downloads in shared profile" feature is enabled or not.
	PrlVmCfg_IsUseMovies (p. 648)	Determines whether the "used movies in shared profile" feature is enabled or not.

	PrlVmCfg_IsUseMusic (p. 648)	Determines whether the "use music in shared profile" feature is enabled or not.
	PrlVmCfg_IsUsePictures (p. 649)	Determines whether the "used pictures in shared profile" feature is enabled or not.
	PrlVmCfg_IsUserDefinedSharedFoldersEnabled (p. 649)	Determines whether the user-defined shared folders are enabled or not.
	PrlVmCfg_IsVNCEncryptionEnabled (p. 650)	Determines if VNC is encrypted.
	PrlVmCfg_Set3DAccelerationMode (p. 650)	Sets 3D graphical acceleration technology mode for the specified virtual machine.
	PrlVmCfg_SetActionOnGuestCrash (p. 651)	%VM_ONLY% Allows to set the on-crash action for the specified virtual machine
	PrlVmCfg_SetActionOnStopMode (p. 652)	Sets action on stop mode.
	PrlVmCfg_SetActionOnWindowClose (p. 652)	Allows to set the action on Virtuozzo Application window close for the specified virtual machine.
	PrlVmCfg_SetAdaptiveHypervisorEnabled (p. 653)	Disables or enables adaptive hypervisor.
	PrlVmCfg_SetAllowSelectBootDevice (p. 653)	Switches on/off select boot device dialog on virtual machine startup.
	PrlVmCfg_SetAppTemplateList (p. 654)	Specify application templates list which will be used in the specified Virtuozzo Virtuozzo Container.
	PrlVmCfg_SetAutoApplyIpOnly (p. 654)	Allows to specify whether a virtual machine should be automatically configured for IP addrs.
	PrlVmCfg_SetAutoCompressEnabled (p. 655)	Enables the Automatic HDD compress feature in a virtual machine.
	PrlVmCfg_SetAutoCompressInterval (p. 656)	Sets the time interval at which compacting virtual disks are doing by Automatic HDD compress.
	PrlVmCfg_SetAutoStart (p. 656)	Allows to sets the automatic startup option for the specified virtual machine.
	PrlVmCfg_SetAutoStartDelay (p. 657)	Sets the time delay that will be used during the virtual machine automatic startup.
	PrlVmCfg_SetAutoStop (p. 657)	Allows to set the automatic shutdown mode for the specified virtual machine.
	PrlVmCfg_SetBackgroundPriority (p. 658)	Sets the virtual machine background processes priority.
	PrlVmCfg_SetBatteryStatusEnabled (p. 658)	Disables or enables battery status.
	PrlVmCfg_SetCapabilitiesMask (p. 659)	Sets a capabilities inside the CT.
	PrlVmCfg_SetCloseAppOnShutdown (p. 659)	Allows to specify whether the Virtuozzo application (such Virtuozzo Desktop) will be closed on the virtual machine shutdown.
	PrlVmCfg_SetConfirmationsList (p. 660)	Determines a list of operations with virtual machine which required administrator confirmation
	PrlVmCfg_SetCpuAccelLevel (p. 661)	Sets CPU acceleration level for the specified virtual machine.
	PrlVmCfg_SetCpuCount (p. 661)	Sets the number of CPUs for the specified virtual machine (the CPUs should be present in the machine).
	PrlVmCfg_SetCpuHotplugEnabled (p. 662)	Sets CPU hotplug mode (enabled or disabled) for the specified virtual machine.
	PrlVmCfg_SetCpuLimit (p. 662)	Sets the CPU usage limit (in percent) for a virtual machine.
	PrlVmCfg_SetCpuLimitEx (p. 663)	Sets limit of CPU usage for the virtual machine in Mhz.
	PrlVmCfg_SetCpuMask (p. 663)	Sets the CPU affinity for a virtual machine.
	PrlVmCfg_SetCpuMode (p. 664)	Sets CPU mode (32 bit or 64 bit) for the specified virtual machine.

	PrlVmCfg_SetCpuUnits (p. 664)	Sets the number of CPU units that will be allocated to a virtual machine.
	PrlVmCfg_SetCustomProperty (p. 665)	Sets the virtual machine custom property information.
	PrlVmCfg_SetDefaultConfig (p. 666)	The PrlVmCfg_SetDefaultConfig function selects a default configuration for a new virtual machine based on the operating system that the virtual machine will be running.
	PrlVmCfg_SetDescription (p. 667)	Sets the virtual machine description.
	PrlVmCfg_SetDisableAPICSign (p. 667)	Allows to specify whether the virtual machine should be using APIC during runtime.
	PrlVmCfg_SetDisableSpeakerSign (p. 668)	Allows to specify whether the virtual machine should be using speaker during runtime.
	PrlVmCfg_SetDiskCacheWriteBack (p. 668)	Sets the virtual machine disk cache write-back option.
	PrlVmCfg_SetDnsServers (p. 669)	Allows to manually assign DNS servers a virtual machine.
	PrlVmCfg_SetEfiEnabled (p. 669)	Sets EFI boot for the virtual machine.
	PrlVmCfg_SetExternalBootDevice (p. 670)	Sets an external boot device system name.
	PrlVmCfg_SetFeaturesMask (p. 671)	Sets a features inside the CT.
	PrlVmCfg_SetForegroundPriority (p. 671)	Sets the virtual machine foreground processes priority.
	PrlVmCfg_SetFreeDiskSpaceRatio (p. 672)	Sets the time free disk space ratio at which compacting virtual disks are doing by Automatic HDD compress.
	PrlVmCfg_SetGuestSharingAutoMount (p. 672)	Sets the guest OS sharing auto-mount option.
	PrlVmCfg_SetGuestSharingEnabled (p. 673)	Enables the guest sharing feature.
	PrlVmCfg_SetGuestSharingEnableSpotlight (p. 674)	Sets the VM virtual disks added to Spotlight search subsystem sign.
	PrlVmCfg_SetHighAvailabilityEnabled (p. 674)	Enables or disables the High Availability feature for a virtual machine.
	PrlVmCfg_SetHighAvailabilityPriority (p. 675)	Sets the priority of the virtual machine in the High Availability Cluster.
	PrlVmCfg_SetHostname (p. 676)	Sets the specified virtual machine hostname.
	PrlVmCfg_SetHostSharingEnabled (p. 676)	Enables host sharing for the specified virtual machine.
	PrlVmCfg_SetIcon (p. 677)	Sets the virtual machine icon.
	PrlVmCfg_SetIoLimit (p. 677)	Sets limit of I/O usage for the virtual machine in Bytes/s.
	PrlVmCfg_SetIoPriority (p. 678)	Sets the specified virtual machine I/O priority.
	PrlVmCfg_SetIopsLimit (p. 678)	Sets the specified virtual machine IOPS limit.
	PrlVmCfg_SetIsolatedVmEnabled (p. 679)	Disables or enables shared camera isolated VM.
	PrlVmCfg_SetLockGuestOnSuspendEnabled (p. 679)	Disables or enables shared camera lock guest on suspend.
	PrlVmCfg_SetLockInFullScreenMode (p. 680)	Enables or disables the lock in screen mode feature in a virtual machine configuration.
	PrlVmCfg_SetLongerBatteryLifeEnabled (p. 681)	Disables or enables longer battery life.
	PrlVmCfg_SetMapSharedFoldersOnLetters (p. 681)	Enables mapping of shared host disks on drive letters for the specified virtual machine.
	PrlVmCfg_SetMemGuaranteeSize (p. 682)	Sets the minimum memory limit which is guaranteed for the virtual machine, in percent of guest RAM.
	PrlVmCfg_SetName (p. 682)	Sets the specified virtual machine name.
	PrlVmCfg_SetNestedVirtualizationEnabled (p. 683)	Disables or enables nested virtualization.
	PrlVmCfg_SetNetfilterMode (p. 683)	Allows to set netfilter mode inside the CT.
	PrlVmCfg_SetNetworkRateList (p. 684)	Add the specified rates list to the Vm configuration.
	PrlVmCfg_SetNodeMask (p. 685)	Sets the NUMA node mask for a virtual machine.

	PrlVmCfg_SetOfflineManagementEnabled (p. 685)	Enables or disables the offline management feature for a virtual machine.
	PrlVmCfg_SetOfflineServices (p. 686)	Allows to manually set offline services that will be available in the virtual machine offline management.
	PrlVmCfg_SetOptimizeModifiersMode (p. 686)	Sets optimize modifiers mode.
	PrlVmCfg_SetOptionsOnGuestCrash (p. 687)	%VM_ONLY% Allows to set the on-crash options for the specified virtual machine
	PrlVmCfg_SetOsResInFullScrMode (p. 687)	Sets the virtual machine OS resolution in full screen mode option.
	PrlVmCfg_SetOsTemplate (p. 688)	Specify the Virtual Machine Operation System template name which will be used to create new Virtuozzo Virtuozzo Container.
	PrlVmCfg_SetOsVersion (p. 688)	Sets the virtual machine guest OS version.
	PrlVmCfg_SetPMUVirtualizationEnabled (p. 689)	Disables or enables PMU virtualization.
	PrlVmCfg_SetProfile (p. 690)	Set one of predefined VM profiles.
	PrlVmCfg_SetRamHotplugEnabled (p. 690)	Sets the virtual machine memory (RAM) hotplug mode (enabled or disabled).
	PrlVmCfg_SetRamSize (p. 691)	Sets the virtual machine memory (RAM) size.
	PrlVmCfg_SetRateBound (p. 691)	Sets the ratebound flag.
	PrlVmCfg_SetResource (p. 692)	Sets barrier and limit for various user beancounters.
	PrlVmCfg_SetScrResEnabled (p. 692)	Enables or disables the additional screen resolution support in a virtual machine.
	PrlVmCfg_SetSearchDomains (p. 693)	Allows to manually set the global search domain list that will be assigned to the Virtuozzo Container.
	PrlVmCfg_SetShareAllHostDisks (p. 694)	Enables sharing of all host disks for the specified virtual machine.
	PrlVmCfg_SetShareClipboard (p. 694)	Enables or disables the clipboard sharing feature.
	PrlVmCfg_SetSharedProfileEnabled (p. 695)	Enables or disables the Shared Profile feature in a virtual machine.
	PrlVmCfg_SetShareUserHomeDir (p. 695)	Enables or disables sharing of the host user home directory in the specified virtual machine.
	PrlVmCfg_SetSmartGuardEnabled (p. 696)	Enables the SmartGuard feature in a virtual machine.
	PrlVmCfg_SetSmartGuardInterval (p. 697)	Sets the time interval at which snapshots are taken by SmartGuard.
	PrlVmCfg_SetSmartGuardMaxSnapshotsCount (p. 697)	Sets the maximum snapshot count, a SmartGuard feature.
	PrlVmCfg_SetSmartGuardNotifyBeforeCreation (p. 698)	Enables or disables notification of automatic snapshot creation, a SmartGuard feature.
	PrlVmCfg_SetSmartMountDVDsEnabled (p. 699)	Enables or disables the SmartMount CD/DVD drives.
	PrlVmCfg_SetSmartMountEnabled (p. 699)	Enables or disables the SmartMount feature.
	PrlVmCfg_SetSmartMountNetworkSharesEnabled (p. 700)	Enables or disables the SmartMount network shares.
	PrlVmCfg_SetSmartMountRemovableDrivesEnabled (p. 700)	Enables or disables the SmartMount removable drives.
	PrlVmCfg_SetStartInDetachedWindowEnabled (p. 701)	Disables or enables start in detached window.
	PrlVmCfg_SetStartLoginMode (p. 701)	Allows to set the automatic startup login mode for the specified virtual machine.

	PrlVmCfg_SetStartUserCreds (p. 702)	Allows to set the automatic startup user login and password for the specified virtual machine.
	PrlVmCfg_SetSwitchOffWindowsLogoEnabled (p. 702)	Disables or enables Windows logo.
	PrlVmCfg_SetSystemFlags (p. 703)	Sets the virtual machine system flags.
	PrlVmCfg_SetTemplateSign (p. 704)	Allows to modify a regular virtual machine to become a template, and vise versa.
	PrlVmCfg_SetTimeSynchronizationEnabled (p. 704)	Enables or disables the time synchronization feature in a virtual machine.
	PrlVmCfg_SetTimeSyncInterval (p. 705)	Allows to set the time interval at which the time in the virtual machine will be synchronized with the time in the host OS.
	PrlVmCfg_SetTimeSyncSmartModeEnabled (p. 706)	Enables or disables the smart time-synchronization mode in a virtual machine.
	PrlVmCfg_SetToolsAutoUpdateEnabled (p. 706)	Enables or disables the Virtuozzo Tools AutoUpdate feature for a virtual machine.
	PrlVmCfg_SetUndoDisksMode (p. 707)	Sets the undo-disks mode for the virtual machine.
	PrlVmCfg_SetUseDefaultAnswers (p. 707)	Enables the use default answers mechanism in a virtual machine.
	PrlVmCfg_SetUseDesktop (p. 708)	Enables or disables the "undo-desktop" feature in the shared profile.
	PrlVmCfg_SetUseDocuments (p. 709)	Lets to set use documents in shared profile enabling sign.
	PrlVmCfg_SetUseDownloads (p. 709)	Lets to set use downloads in shared profile enabling sign.
	PrlVmCfg_SetUseMovies (p. 710)	Lets to set use movies in shared profile enabling sign.
	PrlVmCfg_SetUseMusic (p. 710)	Lets to set use music in shared profile enabling sign.
	PrlVmCfg_SetUsePictures (p. 711)	Lets to set use pictures in shared profile enabling sign.
	PrlVmCfg_SetUserDefinedSharedFoldersEnabled (p. 711)	Enables or disables user-defined shared folders.
	PrlVmCfg_SetUuid (p. 712)	Sets the virtual machine UUID (universally unique ID).
	PrlVmCfg_SetVideoRamSize (p. 712)	Sets the virtual machine video memory size.
	PrlVmCfg_SetVmType (p. 713)	Specify the Virtual Machine type on create action.
	PrlVmCfg_SetVNCHostName (p. 713)	Sets the virtual machine VNC host name.
	PrlVmCfg_SetVNCMode (p. 714)	Sets the virtual machine VNC mode.
	PrlVmCfg_SetVNCPassword (p. 714)	Sets the virtual machine VNC password.
	PrlVmCfg_SetVNCPort (p. 715)	Sets the virtual machine VNC port number.
	PrlVmCfg_SetWindowMode (p. 715)	Sets the virtual machine window mode.

PrlVmCfg_AddDefaultDevice Function

The PrlVmCfg_AddDefaultDevice function automates the task of setting devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_AddDefaultDevice(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hSrvConfig,
    PRL_DEVICE_TYPE deviceType
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314) containing the host machine configuration information. If this parameter is set to PRL_INVALID_HANDLE (p. 1627), the device of the specified type will be configured as pure virtual (i.e. it will not be mapped to a physical device on the host).

deviceType

The type of the device to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OPERATION_FAILED (p. 1333) - unable to set a device of the specified type.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When creating or modifying a virtual machine configuration, one must choose if a device of a particular type will be a pure virtual device or if it will be mapped to a physical device on the host. To make a device pure virtual, set the hSrvConfig parameter value to PRL_INVALID_HANDLE (p. 1627). To map a virtual device to a physical device, obtain an object of type PHT_SERVER_CONFIG (p. 314) and pass it to the function using the hSrvConfig parameter. If multiple physical devices of the specified type exist on the host, the destination physical device will be chosen randomly by the function.

The virtual devices that can be mapped to physical devices include floppy disk drives, CD drives, serial and parallel ports, and sound devices. The devices that can exist only as pure virtual include hard disk drives, network adapters, and USB devices. To determine if a device of particular type must be added to a virtual machine, use the PrlVmCfg_IsDefaultDeviceNeeded (p. 623) function.

Before calling this function, you must execute the PrlVm_BeginEdit (p. 446) function. After the PrlVmCfg_AddDefaultDevice function returns, you must call the PrlVm_Commit (p. 453) to save the changes in the virtual machine configuration.

PrlVmCfg_AddDefaultDeviceEx Function

The PrlVmCfg_AddDefaultDeviceEx function automates the task of setting devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_AddDefaultDeviceEx(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hSrvConfig,
    PRL_DEVICE_TYPE deviceType,
    PRL_HANDLE_PTR phVmDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314) containing the host machine configuration information. If this parameter is set to PRL_INVALID_HANDLE (p. 1627), the device of the specified type will be configured as pure virtual (i.e. it will not be mapped to a physical device on the host).

deviceType

The type of the device to set.

phVmDevice

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OPERATION_FAILED (p. 1333) - unable to set a device of the specified type.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When creating or modifying a virtual machine configuration, one must choose if a device of a particular type will be a pure virtual device or if it will be mapped to a physical device on the host. To make a device pure virtual, set the hSrvConfig parameter value to PRL_INVALID_HANDLE (p. 1627). To map a virtual device to a physical device, obtain an object of type PHT_SERVER_CONFIG (p. 314) and pass it to the function using the hSrvConfig parameter. If multiple physical devices of the specified type exist on the host, the destination physical device will be chosen randomly by the function.

The virtual devices that can be mapped to physical devices include floppy disk drives, CD drives, serial and parallel ports, and sound devices. The devices that can exist only as pure virtual include hard disk drives, network adapters, and USB devices. To determine if a device of particular type must be added to a virtual machine, use the PrlVmCfg_IsDefaultDeviceNeeded (p. 623) function.

Before calling this function, you must execute the PrlVm_BeginEdit (p. 446) function. After the PrlVmCfg_AddDefaultDeviceEx function returns, you must call the PrlVm_Commit (p. 453) to save the changes in the virtual machine configuration.

PrlVmCfg_ApplyConfigSample Function

Specify the Configuration sample file

Syntax

```
PRL_RESULT PrlVmCfg_ApplyConfigSample(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sConfigSample
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. ConfigSample Configurationsample name name.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_CreateBootDev Function

Creates a new handle of type PHT_BOOT_DEVICE (p. 14) and adds it to the specified virtual machine boot device list.

Syntax

```
PRL_RESULT PrlVmCfg_CreateBootDev(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phVmBootDev
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phVmBootDev

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It is the responsibility of the caller to free the handle when it is no longer needed.

PrlVmCfg_CreateShare Function

Creates a new handle of type PHT_SHARE (p. 350) and adds it to the specified virtual machine list of shares.

Syntax

```
PRL_RESULT PrlVmCfg_CreateShare(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phVmShare
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phVmShare

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It is the responsibility of the caller to free the handle when it is no longer needed.

PrlVmCfg_CreateVmDev Function

Creates a new virtual device handle of the specified type.

Syntax

```
PRL_RESULT PrlVmCfg_CreateVmDev(
    PRL_HANDLE hVmCfg,
    PRL_DEVICE_TYPE nDeviceType,
    PRL_HANDLE_PTR phVmDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nDeviceType

The type of the device to create a handle for.

phVmDevice

[out] A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The names of the virtual device handles are prefixed with PHT_VIRTUAL_DEV_ (you may search this documentation for their definitions or see the _PRL_HANDLE_TYPE (p. 996) enumeration for a complete list of handles). It is the responsibility of the user to free the handle when it is no longer needed.

PrlVmCfg_Get3DAccelerationMode Function

Gets 3D graphical acceleration technology mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_Get3DAccelerationMode(
    PRL_HANDLE hVmCfg,
    PRL_VIDEO_3D_ACCELERATION_PTR pn3DAccelerationMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

pn3DAccelerationMode

A pointer to a variable that receives the result. See PRL_VIDEO_3D_ACCELERATION (p. 1100) enum.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_GetAccessRights Function

Obtains a handle of type PHT_ACCESS_RIGHTS (p. 9).

Syntax

```
PRL_RESULT PrlVmCfg_GetAccessRights(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phVmAcl
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phVmAcl

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetActionOnGuestCrash Function

%VM_ONLY%

Determines the on-crash action for the specified virtual machine

Syntax

```
PRL_RESULT PrlVmCfg_GetActionOnGuestCrash(
    PRL_HANDLE hVmCfg,
    PRL_VM_ON_CRASH_ACTION_PTR pnAction
);
```

File

PrlApiVm.h

Parameters

pnAction

[out] A pointer to a variable that receives the result.

hHandle

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetActionOnStopMode Function

Gets action on stop mode.

Syntax

```
PRL_RESULT PrlVmCfg_GetActionOnStopMode(
    PRL_HANDLE hVm,
    PRL_VM_ACTION_ON_STOP_PTR pnMode
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetActionOnWindowClose Function

Determines the action on Virtuozzo Application window close for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetActionOnWindowClose(
    PRL_HANDLE hVmCfg,
    PRL_VM_ACTION_ON_WINDOW_CLOSE_PTR pnActionOnWindowClose
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnActionOnWindowCloase

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetAllDevices Function

Obtains a handles list of all virtual devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetAllDevices(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phDevsList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phDevsList

[out] A pointer to a variable that receives the handle. The type of the handles list is PHT_HANDLES_LIST (p. 108).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a handle.

PrlVmCfg_GetAppTemplateList Function

Returns the list of application templates installed for the specified Container.

Syntax

```
PRL_RESULT PrlVmCfg_GetAppTemplateList(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phAppList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phAppList

[out] A pointer to a variable that receives the PHT_STRINGS_LIST (p. 357) handle containing the list of application templates.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a string object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

This option acceptable for Virtual Machine of PVT_CT only.

PrlVmCfg_GetAutoCompressInterval Function

Determines the interval at which compacting virtual disks are doing by Automatic HDD compress.

Syntax

```
PRL_RESULT PrlVmCfg_GetAutoCompressInterval(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnInterval
```

```
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnInterval

[out] A pointer to a variable the receives the automatic compress interval, in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Automatic HDD compress is a facility that allows to make automatic compress of a virtual machine disks at predefined time intervals. The currently set time interval can be obtained using this function.

See Also

PrlVmCfg_SetAutoCompressInterval (p. 656)

PrlVmCfg_GetAutoStart Function

Determines if the specified virtual machine is set to start automatically on Virtuozzo Service startup.

Syntax

```
PRL_RESULT PrlVmCfg_GetAutoStart(
    PRL_HANDLE hVmCfg,
    PRL_VM_AUTOSTART_OPTION_PTR pnVmAutoStart
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmAutoStart

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetAutoStartDelay Function

Returns the time delay used during the virtual machine automatic startup.

Syntax

```
PRL_RESULT PrlVmCfg_GetAutoStartDelay(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmAutoStartDelay
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmAutoStartDelay

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetAutoStop Function

Determines the mode of the automatic shutdown for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetAutoStop(
    PRL_HANDLE hVmCfg,
    PRL_VM_AUTOSTOP_OPTION_PTR pnVmAutoStop
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmAutoStop

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetBackgroundPriority Function

Determines the specified virtual machine background process priority type.

Syntax

```
PRL_RESULT PrlVmCfg_GetBackgroundPriority(
    PRL_HANDLE hVmCfg,
    PRL_VM_PRIORITY_PTR pnVmBackgroundPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmBackgroundPriority

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetBootDev Function

Obtains a handle of type PHT_BOOT_DEVICE (p. 14) containing information about a specified device from the virtual machine boot device priority list.

Syntax

```
PRL_RESULT PrlVmCfg_GetBootDev(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nBootDevIndex,
    PRL_HANDLE_PTR phVmBootDev
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nBootDevIndex

An index identifying the device (beginning with 0). To determine the number of the devices in the boot device list, use the PrlVmCfg_GetBootDevCount (p. 559) function.

phVmBootDev

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It is the responsibility of the called to free the handle when it is no longer needed.

PrlVmCfg_GetBootDevCount Function

Determines the number of devices in the virtual machine boot device priority list.

Syntax

```
PRL_RESULT PrlVmCfg_GetBootDevCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnBootDevCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnBootDevCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCapabilitiesMask Function

Returns a capabilities mask specified inside the CT.

Syntax

```
PRL_RESULT PrlVmCfg_GetCapabilitiesMask(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnCapMask
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnCapMask

[out] A pointer to a bitmask containing the supported capabilities. For available options see the PRL_CT_CAPMASK (p. 1046) enumeration and CAPDEFAULTMASK (p. 1661) defined in PrlEnums.h as macros.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetCapabilitiesMask (p. 659)

PrlVmCfg_GetConfigValidity Function

The PrlVmCfg_GetConfigValidity function returns an error code.

Syntax

```
PRL_RESULT PrlVmCfg_GetConfigValidity(
    PRL_HANDLE hVmCfg,
    PRL_RESULT_PTR pnErrCode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnErrCode

[out] A pointer to a variable that receives the error code as PRL_RESULT.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The error code indicates the status of the config: config doesn't exist, parsing error, config can be restored and others errors.

PrlVmCfg_GetConfirmationsList Function

Obtains a handles list of operations with virtual machine which required administrator confirmation

Syntax

```
PRL_RESULT PrlVmCfg_GetConfirmationsList(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phConfirmList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phConfirmList

[out] A pointer to a variable that receives the handle. The type of the handle is PHT_OPAQUE_TYPE_LIST (p. 177). (values in list consist in enum PRL_ALLOWED_VM_COMMAND (p. 1036))

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a handle.

See Also

PrlSrv_DisableConfirmationMode (p. 236)() PrlSrv_EnableConfirmationMode (p. 236)() PrlSrv_IsConfirmationModeEnabled (p. 268)() PrlVmCfg_GetConfirmationsList() PrlVmCfg_SetConfirmationsList (p. 660)() PrlDispCfg_GetConfirmationsList (p. 39)() PrlDispCfg_SetConfirmationsList (p. 60)()

PrlVmCfg_GetCpuAccelLevel Function

Determines the virtual machine CPU acceleration level.

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuAccelLevel(
    PRL_HANDLE hVmCfg,
    PRL_VM_ACCELERATION_LEVEL_PTR pnVmCpuAccelLevel
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmCpuAccelLevel

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCpuCount Function

Determines the number of CPUs in the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmCpuCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmCpuCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCpuLimit Function

Determines the CPU usage limit of a virtual machine, in percent.

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuLimit(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR nVmCpuLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmCpuLimit

[out] A pointer to a variable that receives CPU usage limit, in percent.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetCpuLimit (p. 662)

PrlVmCfg_GetCpuLimitEx Function

Determines the specified virtual machine CPU limit.

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuLimitEx(
    PRL_HANDLE hVmCfg,
    PRL_CPULIMIT_DATA_PTR pVmCpuLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmCpuLimit

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCpuMask Function

Returns the specified virtual machine CPU affinity mask

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuMask(
    PRL_HANDLE hVmCfg,
    PRL_STR sMask,
    PRL_UINT32_PTR pnMaskBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sMask

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCpuMode Function

Determines the specified virtual machine CPU mode (32 bit or 64 bit).

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuMode(
    PRL_HANDLE hVmCfg,
    PRL_CPU_MODE_PTR pnVmCpuMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmCpuMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCpuUnits Function

Determines the number of CPU units allocated to a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetCpuUnits(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR nVmCpuUnits
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmCpuUnits

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetCpuUnits (p. 664)

PrlVmCfg_GetCtId Function

Returns the Container id of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetCtId(
    PRL_HANDLE hVmCfg,
    PRL_STR sCtId,
    PRL_UINT32_PTR pnCtIdBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sCtId

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnCtIdBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetCustomProperty Function

Returns the virtual machine custom property information.

Syntax

```
PRL_RESULT PrlVmCfg_GetCustomProperty(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmCustomProperty,
    PRL_UINT32_PTR pnVmCustomPropertyBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmCustomProperty

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string). The custom property information is stored in CDATA format.

pnVmCustomPropertyBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is too small. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If you have some custom data which relates to a virtual machine in some way, you can save it in the virtual machine configuration for future retrieval. This can be any data that your client application requires. The data is stored in CDATA format.

See Also

PrlVmCfg_SetCustomProperty (p. 665)

PrlVmCfg_GetDefaultHddSize Function

The PrlVmCfg_GetDefaultHddSize function returns the default hard disk size for to the specified OS type and version.

Syntax

```
PRL_RESULT PrlVmCfg_GetDefaultHddSize(
    PRL_UINT32 guestOsVersion,
    PRL_UINT32_PTR pnHddSize
);
```

File

PrlApiVm.h

Parameters

guestOsVersion

Operating system version. See PrlOses.h for version definitions.

pnHddSize

[out] A pointer to a variable that receives the hard disk size, in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When creating a virtual machine using a default configuration, the default hard disk is defined for every supported operating system type and version. The default HDD sizes can be retrieved using this function. The OS types are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_TYPE_ (e.g. PVS_GUEST_TYPE_MACOS (p. 1583)). The OS versions can be found in the same file. The names of the macros are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

PrlVmCfg_GetDefaultMemSize Function

The PrlVmCfg_GetDefaultMemSize function returns the default RAM size for to the specified OS type and version.

Syntax

```
PRL_RESULT PrlVmCfg_GetDefaultMemSize(
    PRL_UINT32 guestOsVersion,
```

```
    PRL_UINT32 hostRam,
    PRL_UINT32_PTR pnMemSize
);
```

File

PrlApiVm.h

Parameters

guestOsVersion

Operating system version. See PrlOses.h for type definitions.

hostRam

Optional. The amount of memory (RAM) available on the host system. For some operating system, this parameter may be used to calculate the default the RAM size more precisely.

pnMemSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When creating a virtual machine using a default configuration, the default memory size is defined for every supported operating system type and version. The default sizes can be retrieved using this function. The OS types are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_TYPE_ (e.g. PVS_GUEST_TYPE_MACOS (p. 1583)). The OS versions can be found in the same file. The names of the macros are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

PrlVmCfg_GetDefaultVideoRamSize Function

The PrlVmCfg_GetDefaultVideoRamSize function returns the default video RAM size for the specified OS type and version.

Syntax

```
PRL_RESULT PrlVmCfg_GetDefaultVideoRamSize(
    PRL_UINT32 guestOsVersion,
    PRL_HANDLE hSrvConfig,
    PRL_BOOL bIs3DSupportEnabled,
    PRL_UINT32_PTR pnVideoRamSize
);
```

File

PrlApiVm.h

Parameters

guestOsVersion

Operating system version. See PrlOses.h for type definitions.

hSrvConfig

Optional. The host hardware info object handle (PHT_SERVER_CONFIG (p. 314) type) contains necessary information about host hardware configuration (amount of RAM, host type and etc.). For some operating system, this parameter may be used to calculate the default video RAM size more precisely.

bIs3DSupportEnabled

sign which specifies whether 3D support will be enabled at creating VM. 3D support require at least 128 Mb of video RAM in common case.

pnVideoRamSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When creating a virtual machine using a default configuration, the default video memory size is defined for every supported operating system type and version. The default sizes can be retrieved using this function. The OS types are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_TYPE_ (e.g. PVS_GUEST_TYPE_MACOS (p. 1583)). The OS versions can be found in the same file. The names of the macros are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

PrlVmCfg_GetDescription Function

Returns the specified virtual machine description.

Syntax

```
PRL_RESULT PrlVmCfg_GetDescription(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmDescription,
    PRL_UINT32_PTR pnVmDescriptionBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmDescription

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmDescriptionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetDevByType Function

Obtains a handle to a virtual device in a virtual machine according to the specified device type and index.

Syntax

```
PRL_RESULT PrlVmCfg_GetDevByType(
    PRL_HANDLE hVmCfg,
    PRL_DEVICE_TYPE vmDeviceType,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

vmDeviceType

The device type.

nIndex

An index identifying the device instance (beginning with 0). To determine the number of the devices of the specified type available in the virtual machine, use the PrlVmCfg_GetDevsCountByType (p. 571) function.

phDevice

[out] A pointer to a variable that receives the handle. The type of the handle can be determined using the PrlHandle_GetType (p. 750) function. You should also "know" the handle type based on the device type for which you are obtaining a handle. Search this documentation for topics with PHT_VIRTUAL_DEV_ prefix in their names for the list of the available virtual device handles.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetDevsCountByType (p. 571)

PrlVmCfg_GetDevsCount Function

Determines the total number of devices of any type installed in the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetDevsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR vmdev_handles_count
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

vmdev_handles_count

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetDevsList

PrlVmCfg_GetDevsCountByType Function

Obtains the number of virtual devices of the specified type available in the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetDevsCountByType(
    PRL_HANDLE hVmCfg,
    PRL_DEVICE_TYPE vmDeviceType,
    PRL_UINT32_PTR pnDevicesCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

vmDeviceType

Device type.

pnDevicesCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetDevByType (p. 570)

PrlVmCfg_GetDisplayDev Function

Obtains a handle of type PHT_VIRTUAL_DEV_DISPLAY containing information about a display device in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetDisplayDev(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the display device (beginning with 0). To obtain the number of display devices in a virtual machine, use the PrlVmCfg_GetDisplayDevsCount (p. 572) function.

phDevice

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetDisplayDevsCount (p. 572)

PrlVmCfg_GetDisplayDevsCount Function

Determines the number of display devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetDisplayDevsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnDisplayDevsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnGenericScsiDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetDisplayDev (p. 572)

PrlVmCfg_GetDnsServers Function

Obtains the list of of globally defined DNS servers which is assigned to a Virtuozzo Container.

Syntax

```
PRL_RESULT PrlVmCfg_GetDnsServers(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phDnsServersList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phDnsServersList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle that receives the list of DNS servers.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate strings list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetExternalBootDevice Function

Returns an external boot device system name.

Syntax

```
PRL_RESULT PrlVmCfg_GetExternalBootDevice(
    PRL_HANDLE hVmCfg,
    PRL_STR sSysName,
    PRL_UINT32_PTR pnSysNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sSysName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnSysNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetFeaturesMask Function

Returns a features mask specified inside the CT.

Syntax

```
PRL_RESULT PrlVmCfg_GetFeaturesMask(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnOn,
    PRL_UINT32_PTR pnOff
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnOn

[out] A pointer to a bitmask containing the enabled features. For available options see the PRL_CT_FEATURES (p. 1047) enumeration.

pnOff

[out] A pointer to a bitmask containing the disabled features. For available options see the PRL_CT_FEATURES (p. 1047) enumeration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetCapabilitiesMask (p. 659)

PrlVmCfg_GetFloppyDisk Function

Obtains a handle of type PHT_VIRTUAL_DEV_FLOPPY (p. 381) containing information about the specified floppy disk drive in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetFloppyDisk(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the floppy disk drive instance (beginning with 0). To get the number of floppy disk drives in the virtual machine, use the PrlVmCfg_GetFloppyDisksCount (p. 575) function.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetFloppyDisksCount Function

Determines the number of floppy disk drives in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetFloppyDisksCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnFloppyDisksCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnFloppyDisksCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetForegroundPriority Function

Returns foreground processes priority for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetForegroundPriority(
    PRL_HANDLE hVmCfg,
    PRL_VM_PRIORITY_PTR pnVmForegroundPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmForegroundPriority

[out] A pointer to a variable that receives the result

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetFreeDiskSpaceRatio Function

Determines the free disk space ratio at which compacting virtual disks are doing by Automatic HDD compress.

Syntax

```
PRL_RESULT PrlVmCfg_GetFreeDiskSpaceRatio(
    PRL_HANDLE hVmCfg,
    PRL_DOUBLE_PTR pdFreeDiskSpaceRatio
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pdFreeDiskSpaceRatio

[out] A pointer to a variable the receives the automatic compress space ratio, in percents.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Automatic HDD compress is a facility that allows to make automatic compress of a virtual machine disks at predefined space ratio. The currently set space ratio can be obtained using this function.

See Also

PrlVmCfg_SetFreeDiskSpaceRatio (p. 672)

PrlVmCfg_GetGenericPciDev Function

Obtains a handle of type PHT_VIRTUAL_DEV_GENERIC_PCI containing information about a generic PCI device in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetGenericPciDev(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the generic PCI device (beginning with 0). To obtain the number of generic PCI devices in a virtual machine, use the PrlVmCfg_GetGenericPciDevsCount (p. 578) function.

phDevice

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetGenericPciDevsCount (p. 578)

PrlVmCfg_GetGenericPciDevsCount Function

Determines the number of generic PCI devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetGenericPciDevsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnGenericPciDevsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnGenericPciDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetGenericPciDev (p. 577)

PrlVmCfg_GetGenericScsiDev Function

Obtains a handle of type PHT_VIRTUAL_DEV_GENERIC_SCSI containing information about a generic SCSI device in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetGenericScsiDev(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the generic SCSI device (beginning with 0). To obtain the number of generic SCSI devices in a virtual machine, use the PrlVmCfg_GetGenericScsiDevsCount (p. 579) function.

phDevice

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetGenericScsiDevsCount (p. 579)

PrlVmCfg_GetGenericScsiDevsCount Function

Determines the number of generic SCSI devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetGenericScsiDevsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnGenericScsiDevsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnGenericScsiDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetGenericScsiDev (p. 578)

PrlVmCfg_GetGuestCpuLimitType Function

Determines the specified virtual machine CPU limit type.

Syntax

```
PRL_RESULT PrlVmCfg_GetGuestCpuLimitType(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmGuestCpuLimitType
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmGuestCpuLimitType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetHardDisk Function

Obtains a handle of type PHT_VIRTUAL_DEV_HARD_DISK (p. 381) containing the specified virtual hard disk information.

Syntax

```
PRL_RESULT PrlVmCfg_GetHardDisk(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the hard disk instance (beginning with 0). To get the number of hard disks installed in the virtual machine, use the PrlVmCfg_GetHardDisksCount (p. 581) function.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetHardDisksCount Function

Determines the number of virtual hard disks in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetHardDisksCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnHardDisksCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnHardDisksCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetHighAvailabilityPriority Function

Determines the priority of the virtual machine in the High Availability Cluster.

Syntax

```
PRL_RESULT PrlVmCfg_GetHighAvailabilityPriority(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnPriority

[out] A pointer to a variable the receives the priority value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetHighAvailabilityPriority (p. 675)

PrlVmCfg_GetHomePath Function

Returns the virtual machine home directory name and path.

Syntax

```
PRL_RESULT PrlVmCfg_GetHomePath(
    PRL_HANDLE hVmCfg,
    PRL_STR sHomePath,
    PRL_UINT32_PTR pnHomePathBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sHomePath

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnHomePathBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetHostname Function

Obtains the hostname of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetHostname(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmHostname,
    PRL_UINT32_PTR pnVmHostnameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The pnVmNameBufLength parameter will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetIcon Function

Returns the name of the icon file used by the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetIcon(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmIcon,
    PRL_UINT32_PTR pnVmIconBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmIcon

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmIconBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetIoLimit Function

Determines the specified virtual machine I/O limit.

Syntax

```
PRL_RESULT PrlVmCfg_GetIoLimit(
    PRL_HANDLE hVmCfg,
    PRL_IOLIMIT_DATA_PTR pVmIoLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmIoLimit

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetIoPriority Function

Determines the specified virtual machine I/O priority.

Syntax

```
PRL_RESULT PrlVmCfg_GetIoPriority(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmIoPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmIoPriority

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetIopsLimit Function

Determines the specified virtual machine IOPS limit.

Syntax

```
PRL_RESULT PrlVmCfg_GetIopsLimit(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmIopsLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmIopsLimit

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetLastModifiedDate Function

Returns the date and time when the specified virtual machine was last modified.

Syntax

```
PRL_RESULT PrlVmCfg_GetLastModifiedDate(
```

```
    PRL_HANDLE hVmCfg,
    PRL_STR sVmLastModifiedDate,
    PRL_UINT32_PTR pnVmLastModifiedDateBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmLastModifiedDate

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string). The date is returned using the "yyyy-mm-dd hh:mi:ss" format.

pnVmLastModifiedDateBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Returned time is a local time (i.e. the time value is already converted to local time zone).

PrlVmCfg_GetLastModifierName Function

Returns the name of the user who last modified the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetLastModifierName(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmLastModifierName,
    PRL_UINT32_PTR pnVmLastModifierNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmLastModifierName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmLastModifierNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required

size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetLinkedVmUuid Function

Returns the UUID (universally unique ID) of the parent virtual machine which was used to create current linked cloned virtual machine (if last one is linked clone of some virtual machine - this value will be empty otherwise).

Syntax

```
PRL_RESULT PrlVmCfg_GetLinkedVmUuid(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmUuid,
    PRL_UINT32_PTR pnVmUuidBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetLocation Function

Returns the virtual machine files location type.

Syntax

```
PRL_RESULT PrlVmCfg_GetLocation(
    PRL_HANDLE hVmCfg,
    PRL_VM_LOCATION_PTR pnVmLocation
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmLocation

[out] A pointer to a buffer that receives the result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Please note that this parameter is runtime and read only.

PrlVmCfg_GetMemGuaranteeSize Function

Returns the minimum memory limit which is guaranteed for the virtual machine, in percent of guest RAM.

Syntax

```
PRL_RESULT PrlVmCfg_GetMemGuaranteeSize(
    PRL_HANDLE hVmCfg,
    PRL_MEMGUARANTEE_DATA_PTR pMemGuaranteeSize
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pMemGuaranteeSize

[out] A pointer to a structure that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the memory information is not available in the virtual machine configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetName Function

Returns the name of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetName(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmName,
    PRL_UINT32_PTR pnVmNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetNetAdapter Function

Obtains a handle of type PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399) containing information about a virtual network adapter in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetNetAdapter(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the network adapter instance (beginning with 0). To determine the number of network adapters in the machine, use the PrlVmCfg_GetNetAdaptersCount (p. 590) function.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetNetAdaptersCount Function

Determines the number of network adapters in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetNetAdaptersCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnNetAdaptersCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnNetAdaptersCount

specifies a pointer to the count of devices

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetNetfilterMode Function

Returns netfilter mode specified inside the CT.

Syntax

```
PRL_RESULT PrlVmCfg_GetNetfilterMode(
    PRL_HANDLE hVmCfg,
    PRL_NETFILTER_MODE_PTR pnMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetNetfilterMode (p. 683)

PrlVmCfg_GetNetworkRateList Function

Obtains the list of shaping rated

Syntax

```
PRL_RESULT PrlVmCfg_GetNetworkRateList(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phNetworkRateList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phNetworkRateList

[out] A pointer to the PHT_HANDLES_LIST (p. 108) handle containing the list of handle sof type PHT_NETWORK_RATE (p. 157).

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate the object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetSearchDomains (p. 693)

PrlVmCfg_GetNodeMask Function

Returns the specified virtual machine node mask

Syntax

```
PRL_RESULT PrlVmCfg_GetNodeMask(
    PRL_HANDLE hVmCfg,
    PRL_STR sMask,
    PRL_UINT32_PTR pnMaskBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sMaski

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetOfflineServices Function

Obtains the list of services available in the virtual machine offline management.

Syntax

```
PRL_RESULT PrlVmCfg_GetOfflineServices(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phOfflineServicesList
);
```

File

PrlApiVm.h

Parameters

phOfflineServicesList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle containing the list of offline services.

hVmDev

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate the string list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetOfflineManagementEnabled (p. 685)

PrlVmCfg_SetOfflineServices (p. 686)

PrlVmCfg_GetOpticalDisk Function

Obtains a handle of type PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425) containing information a virtual optical disk in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetOpticalDisk(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the optical disk instance (beginning with 0). To determine the number of optical disks in a virtual machine, use the PrlVmCfg_GetOpticalDisksCount (p. 594) function.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetOpticalDisksCount Function

Determines the number of optical disks in the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetOpticalDisksCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnOpticalDisksCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnOpticalDisksCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetOptimizeModifiersMode Function

Gets optimize modifiers mode.

Syntax

```
PRL_RESULT PrlVmCfg_GetOptimizeModifiersMode(
    PRL_HANDLE hVm,
    PRL_OPTIMIZE_MODIFIERS_MODE_PTR pnMode
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetOptionsOnGuestCrash Function

%VM_ONLY%

Determines the on-crash options for the specified virtual machine

Syntax

```
PRL_RESULT PrlVmCfg_GetOptionsOnGuestCrash(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnOptions
);
```

File

PrlApiVm.h

Parameters

pnOptions

[out] A pointer to a variable that recieves additional options.

hHandle

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetOsTemplate Function

Returns Operation System template name the Virtuozzo Virtuozzo Container were created on.

Syntax

```
PRL_RESULT PrlVmCfg_GetOsTemplate(
    PRL_HANDLE hVmCfg,
    PRL_STR sOsTemplate,
    PRL_UINT32_PTR pnBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sOsTemplate

[out] A pointer to a buffer that receives the result.

pnBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetOsType Function

Returns the type of the operating system that the specified virtual machine is running.

Syntax

```
PRL_RESULT PrlVmCfg_GetOsType(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmOsType
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmOsType

[out] A pointer to a variable that receives the OS type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - VM - the OS information is not present in the virtual machine configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The OS types are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_TYPE_ (e.g. PVS_GUEST_TYPE_MACOS (p. 1583)).

PrlVmCfg_GetOsVersion Function

Returns the version of the operating system that the specified virtual machine is running.

Syntax

```
PRL_RESULT PrlVmCfg_GetOsVersion(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmOsVersion
```

```
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmOsVersion

[out] A pointer to a variable that receives the OS version.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - VM - the OS information is not present in the virtual machine configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The OS versions are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

PrlVmCfg_GetParallelPort Function

Obtains a handle of type PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425) containing information about a virtual printer port in the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetParallelPort(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the printer port instance (beginning with 0). To determine the number of printer ports in a virtual machine, use the PrlVmCfg_GetParallelPortsCount (p. 598) function.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetParallelPortsCount Function

Determines the number of virtual printer ports in the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetParallelPortsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnParallelPortsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnParallelPortsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetProfile Function

Get current VM profile.

Syntax

```
PRL_RESULT PrlVmCfg_GetProfile(
    PRL_HANDLE hVm,
    PRL_VIRTUAL_MACHINE_PROFILE_PTR pnVmProfile
);
```

File

PrlApiVm.h

Parameters

pnVmProfile

[out] A predefined settings profile. See the PRL_VIRTUAL_MACHINE_PROFILE (p. 1100) enumeration for available options.

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Use PrlVmCfg_IsCustomProfile (p. 623) to detect if settings were modified after profile application.

PrlVmCfg_GetRamSize Function

Returns the virtual machine memory (RAM) size, in megabytes.

Syntax

```
PRL_RESULT PrlVmCfg_GetRamSize(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmRamSize
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmRamSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the memory information is not available in the virtual machine configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetResource Function

Returns beancounters barrier and limit for specified resource id.

Syntax

```
PRL_RESULT PrlVmCfg_GetResource(
    PRL_HANDLE hVmCfg,
    PRL_CT_RESOURCE nResourceId,
    PRL_UINT64_PTR pnBarrirer,
    PRL_UINT64_PTR pnLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. nResourceId The resource id (Only PCR_SWAPPAGES supported for now). pnBarrirer [out]The resource barrier. pnLimit [out]The resource limit

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_PARAM_NOT_FOUND (p. 1335) - resource not found. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

were created on.

PrlVmCfg_GetSearchDomains Function

Obtains the list of globally defined search domains that will be assigned to a virtual machine on the guest operating system side.

Syntax

```
PRL_RESULT PrlVmCfg_GetSearchDomains(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phSearchDomainsList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phSearchDomainsList

[out] A pointer to the PHT_STRINGS_LIST (p. 357) handle containing the list of search domains.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate the string list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetSearchDomains (p. 693)

PrlVmCfg_GetSerialPort Function

Obtains a handle of type PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425) containing information about a serial port in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetSerialPort(
```

```
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the serial port instance.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetSerialPortsCount Function

Determines the number of serial ports in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetSerialPortsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnSerialPortsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnSerialPortsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetServerHost Function

Returns the hostname of the machine hosting the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetServerHost(
    PRL_HANDLE hVmCfg,
    PRL_STR sServerHost,
    PRL_UINT32_PTR pnServerHostBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnServerHostBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

unnamsServerHosted_param

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetServerUuid Function

Returns the UUID of the machine hosting the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetServerUuid(
    PRL_HANDLE hVmCfg,
    PRL_STR sServerUuid,
    PRL_UINT32_PTR pnServerUuidBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sServerUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnServerUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetShare Function

Obtains a handle of type PHT_SHARE (p. 350) containing information about a shared folder.

Syntax

```
PRL_RESULT PrlVmCfg_GetShare(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nShareIndex,
    PRL_HANDLE_PTR phVmShare
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nShareIndex

An index identifying the shared folder (beginning with 0). To determine the number of shared folders in a virtual machine, use the PrlVmCfg_GetSharesCount (p. 604) function.

phVmShare

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A shared folder is a folder in the host OS which is shared with a given virtual machine. To obtain its information, first determine the number of the available shared folders using the PrlVmCfg_GetSharesCount (p. 604) function. Then iterate through the list using the iteration number as the nShareIndex parameter value.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p.

471) function.

PrlVmCfg_GetSharesCount Function

Determines the number of shared folders in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetSharesCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnSharesCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnSharesCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A shared folder is a folder in the host OS which is shared with a given virtual machine. To obtain its information, first determine the number of the available shared folders using the PrlVmCfg_GetSharesCount function. Then obtain information about an individual folder using the PrlVmCfg_GetShare (p. 603) function.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_GetSmartGuardInterval Function

Determines the interval at which snapshots are taken by SmartGuard.

Syntax

```
PRL_RESULT PrlVmCfg_GetSmartGuardInterval(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnInterval
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnInterval

[out] A pointer to a variable the receives the snapshot creation interval, in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is a facility that allows to make automatic snapshots of a virtual machine at predefined time intervals. The currently set time interval can be obtained using this function.

See Also

PrlVmCfg_SetSmartGuardInterval (p. 697)

PrlVmCfg_GetSmartGuardMaxSnapshotsCount Function

Determines the maximum snapshot count, a SmartGuard setting.

Syntax

```
PRL_RESULT PrlVmCfg_GetSmartGuardMaxSnapshotsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnMaxSnapshotsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is a facility that allows to make automatic snapshots of a virtual machine at predefined time intervals. The snapshot count setting specifies how many snapshots taken by SmartGuard will be kept on the hard drive (maximum is 100). As soon as the maximum number is exceeded, the oldest snapshot is deleted.

PrlVmCfg_GetSoundDev Function

Obtains a handle of type PHT_VIRTUAL_DEV_SOUND (p. 427) containing information about a sound device in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetSoundDev(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the sound device (beginning with 0). To obtain the number of sound devices in a virtual machine, use the PrlVmCfg_GetSoundDevsCount (p. 606) function.

phDevice

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetSoundDevsCount Function

Determines the number of sound devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetSoundDevsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnSoundDevsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnSoundDevsCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetStartLoginMode Function

Returns the automatic startup login mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetStartLoginMode(
    PRL_HANDLE hVmCfg,
    PRL_VM_START_LOGIN_MODE_PTR pnVmStartLoginMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmStartLoginMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetStartUserLogin Function

Returns the user name used during the specified virtual machine automatic startup.

Syntax

```
PRL_RESULT PrlVmCfg_GetStartUserLogin(
    PRL_HANDLE hVmCfg,
    PRL_STR sStartUserLogin,
    PRL_UINT32_PTR pnStartUserLoginBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. the virtual machine.

sStartUserLogin

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnStartUserLoginBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetSystemFlags Function

Returns the virtual machine system flags.

Syntax

```
PRL_RESULT PrlVmCfg_GetSystemFlags(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmSystemFlags,
    PRL_UINT32_PTR pnVmSystemFlagsBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmSystemFlags

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmSystemFlagsBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetTimeSyncInterval Function

Obtains the time synchronization interval between the host and a guest OS.

Syntax

```
PRL_RESULT PrlVmCfg_GetTimeSyncInterval(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnTimeSyncInterval
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnTimeSyncInterval

[out] A pointer to a variable that receive the time synchronization value, in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

See PrlVmCfg_SetTimeSynchronizationEnabled (p. 704) for more information.

See Also

PrlVmCfg_SetTimeSyncInterval (p. 705)

PrlVmCfg_GetUndoDisksMode Function

Determines the current undo-disks mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetUndoDisksMode(
    PRL_HANDLE hVmCfg,
    PRL_UNDO_DISKS_MODE_PTR pnUndoDisksMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnUndoDisksMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetUptime Function

Returns the virtual machine uptime in seconds since some datetime (see PrlVmCfg_GetUptimeStartDate (p. 610)).

Syntax

```
PRL_RESULT PrlVmCfg_GetUptime(
    PRL_HANDLE hVmCfg,
    PRL_UINT64_PTR pnVmUptime
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmUptime

[out] The buffer for storing result.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully. See also PrlVmCfg_GetUptimeStartDate (p. 610) PrlStat_GetOsUptime (p. 363)

PrlVmCfg_GetUptimeStartDate Function

Returns the date and time when uptime counter calculation was started for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetUptimeStartDate(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmUptimeStartDate,
    PRL_UINT32_PTR pnVmUptimeStartDateBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmUptimeStartDate

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string). The date is returned using the "yyyy-mm-dd hh:mi:ss" format.

pnVmUptimeStartDateBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully. See also PrlVmCfg_GetUptime (p. 610)

Remarks

Returned time is a local time (i.e. the time value is already converted to local time zone).

PrlVmCfg_GetUsbDevice Function

Obtains a handle of type PHT_VIRTUAL_DEV_USB_DEVICE (p. 430) containing information about a USB device in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetUsbDevice(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nIndex,
    PRL_HANDLE_PTR phDevice
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nIndex

An index identifying the USB device (beginning with 0). To determine the number of USB devices in a virtual machine, use the PrlVmCfg_GetUsbDevicesCount (p. 612) function.

phDevice

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetUsbDevicesCount Function

Determines the number of USB devices in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetUsbDevicesCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnUsbDevicesCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnUsbDevicesCount

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetUuid Function

Returns the UUID (universally unique ID) of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetUuid(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmUuid,
    PRL_UINT32_PTR pnVmUuidBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmUuid

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmUuidBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVideoRamSize Function

Returns the video memory size of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetVideoRamSize(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmVideoRamSize
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmVideoRamSize

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the video memory size information is missing in the virtual machine configuration.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVmInfo Function

Obtains a handle of type PHT_VM_INFO (p. 719) containing the specified virtual machine state and access rights information.

Syntax

```
PRL_RESULT PrlVmCfg_GetVmInfo(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE_PTR phVmInfo
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phVmInfo

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to allocate object

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Possible values are:

PrlVmCfg_GetVmType Function

Get the Virtual Machine type.

Syntax

```
PRL_RESULT PrlVmCfg_GetVmType(
    PRL_HANDLE hVmCfg,
    PRL_VM_TYPE_PTR pnType
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVNCHostName Function

Returns the VNC hostname of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetVNCHostName(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmRemoteDisplayHostName,
    PRL_UINT32_PTR pnVmRemoteDisplayHostNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmRemoteDisplayHostName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmRemoteDisplayHostNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVNCMode Function

Returns the VNC mode of the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetVNCMode(
    PRL_HANDLE hVmCfg,
    PRL_VM_REMOTE_DISPLAY_MODE_PTR pnVmRemoteDisplayMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmRemoteDisplayMode

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVNCPassword Function

Returns the VNC password for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetVNCPassword(
    PRL_HANDLE hVmCfg,
    PRL_STR sVmRemoteDisplayPassword,
    PRL_UINT32_PTR pnVmRemoteDisplayPasswordBufLength
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sVmRemoteDisplayPassword

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnVmRemoteDisplayPasswordBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVNCPort Function

Returns the VNC port number for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetVNCPort(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmRemoteDisplayPort
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmRemoteDisplayPort

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetVNCWebSocketPort Function

Returns the VNC WebSocket port number for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_GetVNCWebSocketPort(
    PRL_HANDLE hVmCfg,
    PRL_UINT32_PTR pnVmRemoteDisplayWebSocketPort
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pnVmRemoteDisplayPort

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_GetWindowMode Function

Returns the current window mode the specified virtual machine is in.

Syntax

```
PRL_RESULT PrlVmCfg_GetWindowMode(
    PRL_HANDLE hVmCfg,
    PRL_VM_WINDOW_MODE_PTR pnVmWindowMode
);
```

File

PrlApiVm.h

Parameters

pnVmWindowMode

[out] A pointer to a variable that receives the result.

hVm

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsAdaptiveHypervisorEnabled Function

Determines disable or enable adaptive hypervisor.

Syntax

```
PRL_RESULT PrlVmCfg_IsAdaptiveHypervisorEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsAllowSelectBootDevice Function

Determines whether select boot device should be shown on virtual machine startup.

Syntax

```
PRL_RESULT PrlVmCfg_IsAllowSelectBootDevice(
    PRL_HANDLE hVmCfg,
```

```
    PRL_BOOL_PTR pbAllowed
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbAllowed

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that select boot device will be show. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetAllowSelectBootDevice (p. 653)

PrlVmCfg_IsAutoApplyIpOnly Function

Determines if the Guest configured to automatically apply IP addrs network settings inside guest.

Syntax

```
PRL_RESULT PrlVmCfg_IsAutoApplyIpOnly(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbAutoApplyIpOnly
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual network adapter.

pbAutoApplyIpOnly

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the adapter is configured to automatically apply network settings. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsAutoCompressEnabled Function

Determines whether the Automatic HDD compress feature is enabled or not in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsAutoCompressEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that Automatic HDD compress is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Automatic HDD compress is feature that allows to make automatic compacting of a virtual machine disks at predefined time intervals. The feature recommends Virtuozzo Tools installed in a virtual machine.

See Also

PrlVmCfg_SetAutoCompressEnabled (p. 655)

PrlVmCfg_IsBatteryStatusEnabled Function

Determines disable or enable battery status.

Syntax

```
PRL_RESULT PrlVmCfg_IsBatteryStatusEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsCloseAppOnShutdown Function

Determines whether the Virtuozzo application (such as Virtuozzo Desktop) will be automatically closed on the virtual machine shutdown.

Syntax

```
PRL_RESULT PrlVmCfg_IsCloseAppOnShutdown(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmCloseAppOnShutdown
);
```

File

PrlApiVm.h

Parameters

pbVmCloseAppOnShutdown

[out] A pointer to a variable that receives the result. PLR_TRUE indicates that applications will be closed. PRL_FALSE (p. 1626) indicates otherwise.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - operation successfully completed.

PrlVmCfg_IsConfigInvalid Function

The PrlVmCfg_IsConfigInvalid function is used to check the

- error code of PrlVmCfg_GetConfigValidity (p. 560) function that
- indicates the invalid status of the config:
- config doesn't exist, parsing error, invalid server uuid,
- invalid VM uuid.
- Parameters
- nErrCode : The nErrCode value from PrlVmCfg_GetConfigValidity (p. 560)()

- pbInvalid : [out] validity sign.

Syntax

```
PRL_RESULT PrlVmCfg_IsConfigInvalid(
    PRL_RESULT nErrCode,
    PRL_BOOL_PTR pbInvalid
);
```

File

PrlApiVm.h

PrlVmCfg_IsCpuHotplugEnabled Function

Determines whether the CPU hotplug is available in the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsCpuHotplugEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmCpuHotplugEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmCpuHotplugEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsCpuVtxEnabled Function

Determines whether the x86 virtualization (such as Vt-x) is available in the virtual machine CPU.

Syntax

```
PRL_RESULT PrlVmCfg_IsCpuVtxEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmCpuVtxEnabled
);
```

File

PrlApiVm.h

Parameters

pbVmCpuVtxEnabled

A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the x86 virtualization is available. PRL_FALSE (p. 1626) indicates otherwise.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_NO_DATA (p. 1324) - virtual machine configuration does not have necessary data.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_IsCustomProfile Function

Determines were settings modified after profile was applied or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsCustomProfile(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbCustom
);
```

File

PrlApiVm.h

Parameters

pbCustom

[out] A pointer to variable that indicates that settings were modified after profile was applied.

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsDefaultDeviceNeeded Function

Determines whether the virtual machine requires a default device of the specified type if it's going to run an operating system of the specified type.

Syntax

```
PRL_RESULT PrlVmCfg_IsDefaultDeviceNeeded(
    PRL_UINT32 guestOsVersion,
    PRL_DEVICE_TYPE deviceType,
    PRL_BOOL_PTR pbNeeded
);
```

File

PrlApiVm.h

Parameters

guestOsVersion

Target OS version. The OS versions are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

deviceType

Device type.

pbNeeded

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the device of specified type is required. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The function accepts the target operating system type, version, and a device type identifier. Using these parameters, it determines if the specified device is required in order for the projected virtual machine to run properly. If the device is required, you must add it to the virtual machine using the PrlVmCfg_AddDefaultDevice (p. 547) function. If the device is not required, you don't have to add it (i.e. it will not affect the operation of a virtual machine).

PrlVmCfg_IsDisableAPIC Function

Determines whether the APIC is enabled or disabled during the virtual machine runtime.

Syntax

```
PRL_RESULT PrlVmCfg_IsDisableAPIC(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbDisableAPIC
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbDisableAPIC

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the APIC is disabled. PBL_FALSE indicates

otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsDisableSpeaker Function

Determines whether the speaker is enabled or disabled during the virtual machine runtime.

Syntax

```
PRL_RESULT PrlVmCfg_IsDisableSpeaker(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbDisableSpeaker
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbDisableSpeaker

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the speaker is disabled. PBL_FALSE indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsDiskCacheWriteBack Function

Determines if disk cache write-back is enabled in the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsDiskCacheWriteBack(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmDiskCacheWriteBack
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmDiskCacheWriteBack

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that cache write-back is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetDiskCacheWriteBack (p. 668)

PrlVmCfg_IsEfiEnabled Function

Determines whether EFI boot is switched on.

Syntax

```
PRL_RESULT PrlVmCfg_IsEfiEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEfiEnabled
);
```

File

PrlApiVm.h

Parameters

pbEfiEnabled

A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the EFI boot is switched on. PRL_FALSE (p. 1626) indicates otherwise.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_IsGuestSharingAutoMount Function

Determines if host shared folders are mounted automatically in a given virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsGuestSharingAutoMount(
    PRL_HANDLE hVmCfg,
```

```
    PRL_BOOL_PTR pbVmGuestSharingAutoMount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmGuestSharingAutoMount

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that shares are mounted automatically. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If shares are mounted automatically, then shared folders will be mapped and appear in the guest OS on startup.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsGuestSharingEnabled Function

Determines if guest sharing is enabled (the guest OS disk drives are visible in the host OS).

Syntax

```
PRL_RESULT PrlVmCfg_IsGuestSharingEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmGuestSharingEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmGuestSharingEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that guest sharing is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsGuestSharingEnableSpotlight Function

Determines if the virtual disks in a virtual machine will be added to Spotlight search subsystem (Mac OS X feature).

Syntax

```
PRL_RESULT PrlVmCfg_IsGuestSharingEnableSpotlight(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmGuestSharingEnableSpotlight
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmGuestSharingEnableSpotlight

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that virtual disks are added to Spotlight search subsystem. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsHighAvailabilityEnabled Function

Determines whether the High Availability feature is enabled or not for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsHighAvailabilityEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that High Availability is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetHighAvailabilityEnabled (p. 674)

PrlVmCfg_IsHostSharingEnabled Function

Determines if host sharing is enabled (host shared folders are visible in the guest OS).

Syntax

```
PRL_RESULT PrlVmCfg_IsHostSharingEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmHostSharingEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmHostSharingEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that sharing is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsIsolatedVmEnabled Function

Determines disable or enable isolated VM.

Syntax

```
PRL_RESULT PrlVmCfg_IsIsolatedVmEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsLockGuestOnSuspendEnabled Function

Determines disable or enable lock guest on suspend.

Syntax

```
PRL_RESULT PrlVmCfg_IsLockGuestOnSuspendEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsLockInFullScreenMode Function

Determines whether the lock in screen mode sign is set or not in a virtual machine configuration.

Syntax

```
PRL_RESULT PrlVmCfg_IsLockInFullScreenMode(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbValue
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbValue

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Lock in screen mode is feature of Virtuozzo Player product that allows to mandatory apply full screen mode for Virtuozzo GUI until guest OS started and go to the windowed mode just after guest OS will be stopped.

See Also

PrlVmCfg_SetLockInFullScreenMode (p. 680)

PrlVmCfg_IsLongerBatteryLifeEnabled Function

Determines disable or enable longer battery life.

Syntax

```
PRL_RESULT PrlVmCfg_IsLongerBatteryLifeEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsMapSharedFoldersOnLetters Function

Determines whether host disks shared with the guest Windows OS will be mapped to drive letters.

Syntax

```
PRL_RESULT PrlVmCfg_IsMapSharedFoldersOnLetters(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbMapSharedFoldersOnLetters
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbMapSharedFoldersOnLetters

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that host disks will be mapped to drive letters. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsNestedVirtualizationEnabled Function

Determines disable or enable nested virtualization.

Syntax

```
PRL_RESULT PrlVmCfg_IsNestedVirtualizationEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsOfflineManagementEnabled Function

Determines whether the offline management feature is enabled for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsOfflineManagementEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that offline management is available. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetOfflineManagementEnabled (p. 685)

PrlVmCfg_IsOsResInFullScrMode Function

Determines wether the virtual machine OS resolution is in full screen mode.

Syntax

```
PRL_RESULT PrlVmCfg_IsOsResInFullScrMode(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmOsResInFullScrMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmOsResInFullScrMode

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates full screen mode. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsPMUVirtualizationEnabled Function

Determines disable or enable PMU virtualization.

Syntax

```
PRL_RESULT PrlVmCfg_IsPMUVirtualizationEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsRamHotplugEnabled Function

Determines whether the memory (RAM) hotplug is available in the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsRamHotplugEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmRamHotplugEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmRamHotplugEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine object does not contain the necessary data.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsRateBound Function

Determines whether the ratebound is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsRateBound(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsScrResEnabled Function

Determines if additional screen resolution support is enabled in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsScrResEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmScrResEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmScrResEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsShareAllHostDisks Function

Determines whether all host disks will be present at guest OS as shares.

Syntax

```
PRL_RESULT PrlVmCfg_IsShareAllHostDisks(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbShareAllHostDisks
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbShareAllHostDisks

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that all host disks will be. shared into guest OS. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsShareClipboard Function

Determines whether the clipboard sharing feature is enabled or disabled in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsShareClipboard(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates that it is disabled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When clipboard sharing is enabled, you can copy and paste text and files from the host to a virtual machine or vise versa, or from one virtual machine to another. This feature requires Virtuozzo Tools installed in a virtual machine.

PrlVmCfg_IsSharedProfileEnabled Function

Determines whether the Shared Profile feature is enabled in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsSharedProfileEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the shred profile is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsSharedProfileEnabled

PrlVmCfg_IsShareUserHomeDir Function

Determines whether the host user home directory will be available in the guest OS as a share.

Syntax

```
PRL_RESULT PrlVmCfg_IsShareUserHomeDir(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbShareUserHomeDir
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbShareUserHomeDir

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that host user home directory will be shared with the guest OS. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_IsSmartGuardEnabled Function

Determines whether the SmartGuard feature is enabled or not in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsSmartGuardEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that SmartGuard is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is feature that allows to make automatic snapshots of a virtual machine at predefined time intervals. The feature requires Virtuozzo Tools installed in a virtual machine.

See Also

PrlVmCfg_SetSmartGuardEnabled (p. 696)

PrlVmCfg_IsSmartGuardNotifyBeforeCreation Function

Determines whether the user will be notified on automatic snapshot creation, a SmartGaurd setting.

Syntax

```
PRL_RESULT PrlVmCfg_IsSmartGuardNotifyBeforeCreation(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbNotifyBeforeCreation
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbNotifyBeforeCreation

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the user will be automatically notified. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

See PrlVmCfg_SetSmartGuardNotifyBeforeCreation (p. 698) for more information.

See Also

PrlVmCfg_SetSmartGuardNotifyBeforeCreation (p. 698)

PrlVmCfg_IsSmartMountDVDsEnabled Function

Determines whether the SmartMount CD/DVD drives is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsSmartMountDVDsEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsSmartMountEnabled Function

Determines whether the SmartMount feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsSmartMountEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsSmartMountNetworkSharesEnabled Function

Determines whether the SmartMount network shares is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsSmartMountNetworkSharesEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsSmartMountRemovableDrivesEnabled Function

Determines whether the SmartMount removable drives is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsSmartMountRemovableDrivesEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsStartInDetachedWindowEnabled Function

Determines disable or enable start in detached window.

Syntax

```
PRL_RESULT PrlVmCfg_IsStartInDetachedWindowEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsSwitchOffWindowsLogoEnabled Function

Determines disable or enable Windows logo.

Syntax

```
PRL_RESULT PrlVmCfg_IsSwitchOffWindowsLogoEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL_PTR pbEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnable

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsTemplate Function

Determines if the specified virtual machine object contains a virtual machine or a template configuration information.

Syntax

```
PRL_RESULT PrlVmCfg_IsTemplate(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmIsTemplate
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmIsTemplate

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the machine is a template. PBL_FALE indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If the object contains a template configuration, it can be used to create new virtual machines from it. See PrlVm_Clone (p. 450) for the information on how to create virtual machines from templates.

PrlVmCfg_IsTimeSynchronizationEnabled Function

Determines whether the time synchronization feature is enabled in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsTimeSynchronizationEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the time synchronization is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetTimeSynchronizationEnabled (p. 704)

PrlVmCfg_IsTimeSyncSmartModeEnabled Function

Determines whether the smart time synchronization is enabled in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsTimeSyncSmartModeEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

A pointer to a variable that receive the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

For more information, see PrlVmCfg_SetTimeSyncSmartModeEnabled (p. 706).

PrlVmCfg_IsToolsAutoUpdateEnabled Function

Enables or disables the Virtuozzo Tools AutoUpdate feature for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsToolsAutoUpdateEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable the AutoUpdate feature. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUseDefaultAnswers Function

Determines whether the use default answers mechanism active for the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_IsUseDefaultAnswers(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbUseDefaultAnswers
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbUseDefaultAnswers

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that mehanism is active. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_SetUseDefaultAnswers (p. 707)

PrlVmCfg_IsUseDesktop Function

Determines whether the "use desktop in shared profile" feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUseDesktop(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUseDocuments Function

Determines whether "use documents in shared profile" feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUseDocuments(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUseDownloads Function

Determines whether the "used downloads in shared profile" feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUseDownloads(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUseMovies Function

Determines whether the "used movies in shared profile" feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUseMovies(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUseMusic Function

Determines whether the "use music in shared profile" feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUseMusic(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUsePictures Function

Determines whether the "used pictures in shared profile" feature is enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUsePictures(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsUserDefinedSharedFoldersEnabled Function

Determines whether the user-defined shared folders are enabled or not.

Syntax

```
PRL_RESULT PrlVmCfg_IsUserDefinedSharedFoldersEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbEnabled
```

```
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer to buffer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_IsVNCEncryptionEnabled Function

Determines if VNC is encrypted.

Syntax

```
PRL_RESULT PrlVmCfg_IsVNCEncryptionEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL_PTR pbVmRemoteDisplayEncryptionEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pbVmRemoteDisplayEncryptionEnabled

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the feature is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_Set3DAccelerationMode Function

Sets 3D graphical acceleration technology mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_Set3DAccelerationMode(
    PRL_HANDLE hVmCfg,
    PRL_VIDEO_3D_ACCELERATION n3DAccelerationMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

n3DAccelerationMode

Setting value. See PRL_VIDEO_3D_ACCELERATION (p. 1100) enum.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmCfg_SetActionOnGuestCrash Function

%VM_ONLY%

Allows to set the on-crash action for the specified virtual machine

Syntax

```
PRL_RESULT PrlVmCfg_SetActionOnGuestCrash(
    PRL_HANDLE hVmCfg,
    PRL_VM_ON_CRASH_ACTION nAction
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nAction

The on-crash action to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetActionOnStopMode Function

Sets action on stop mode.

Syntax

```
PRL_RESULT PrlVmCfg_SetActionOnStopMode(
    PRL_HANDLE hVm,
    PRL_VM_ACTION_ON_STOP nMode
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nMode

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetActionOnWindowClose Function

Allows to set the action on Virtuozzo Application window close for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetActionOnWindowClose(
    PRL_HANDLE hVmCfg,
    PRL_VM_ACTION_ON_WINDOW_CLOSE nActionOnWindowClose
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nActionOnWindowClose

Action on Virtuozzo window close to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetAdaptiveHypervisorEnabled Function

Disables or enables adaptive hypervisor.

Syntax

```
PRL_RESULT PrlVmCfg_SetAdaptiveHypervisorEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetAllowSelectBootDevice Function

Switches on/off select boot device dialog on virtual machine startup.

Syntax

```
PRL_RESULT PrlVmCfg_SetAllowSelectBootDevice(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bAllowed
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bAllowed

Action type. Specify PRL_TRUE (p. 1652) to enable select boot device dialog. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsAllowSelectBootDevice (p. 618)

PrlVmCfg_SetAppTemplateList Function

Specify application templates list which will be used in the specified Virtuozzo Virtuozzo Container.

Syntax

```
PRL_RESULT PrlVmCfg_SetAppTemplateList(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hAppList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hAppList

A handle of type PHT_STRINGS_LIST (p. 357) that specifies the list of application templates

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

This option acceptable for Virtual Machine of PVT_CT only.

PrlVmCfg_SetAutoApplyIpOnly Function

Allows to specify whether a virtual machine should be automatically configured for IP addrs.

Syntax

```
PRL_RESULT PrlVmCfg_SetAutoApplyIpOnly(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bAutoApplyIpOnly
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual network adapter.

bAutoApplyIpOnly

Specify PRL_TRUE (p. 1652) to configure the adapter automatically. Specified PRL_FALSE (p. 1626) to configure the network settings manually in a virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

In comparison with PrlVmDevNet_SetAutoApply (p. 413)(), dns servers and searchdomains settings will be skipped.

Notes

for changes to take effect, Virtuozzo Tools must be installed in a target virtual machine.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

PrlVmCfg_SetAutoCompressEnabled Function

Enables the Automatic HDD compress feature in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetAutoCompressEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable Automatic HDD compress. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Automatic HDD compress is feature that allows to make automatic compress of a virtual machine disks at predefined time intervals. The feature recommends Virtuozzo Tools installed in a virtual machine.

See Also

PrlVmCfg_SetAutoCompressdInterval PrlVmCfg_SetFreeDiskSpaceRatio (p. 672)

PrlVmCfg_SetAutoCompressInterval Function

Sets the time interval at which compacting virtual disks are doing by Automatic HDD compress.

Syntax

```
PRL_RESULT PrlVmCfg_SetAutoCompressInterval(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nInterval
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nInterval

The time interval to set, in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Automatic HDD compress is feature that allows to make automatic compress of a virtual machine disks at predefined time intervals. This function allows to set the interval at which the automatic compress are taken.

See Also

PrlVmCfg_GetAutoCompressInterval (p. 555)

PrlVmCfg_SetAutoStart Function

Allows to sets the automatic startup option for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetAutoStart(
    PRL_HANDLE hVmCfg,
    PRL_VM_AUTOSTART_OPTION nVmAutoStart
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmAutoStart

The automatic startup option to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetAutoStartDelay Function

Sets the time delay that will be used during the virtual machine automatic startup.

Syntax

```
PRL_RESULT PrlVmCfg_SetAutoStartDelay(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmAutoStartDelay
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmAutoStartDelay

Time delay to set, in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetAutoStop Function

Allows to set the automatic shutdown mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetAutoStop(
    PRL_HANDLE hVmCfg,
    PRL_VM_AUTOSTOP_OPTION nVmAutoStop
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmAutoStop

Automatic shutdown mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetBackgroundPriority Function

Sets the virtual machine background processes priority.

Syntax

```
PRL_RESULT PrlVmCfg_SetBackgroundPriority(
    PRL_HANDLE hVmCfg,
    PRL_VM_PRIORITY nVmBackgroundPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmBackgroundPriority

The priority level to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetBatteryStatusEnabled Function

Disables or enables battery status.

Syntax

```
PRL_RESULT PrlVmCfg_SetBatteryStatusEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetCapabilitiesMask Function

Sets a capabilities inside the CT.

Syntax

```
PRL_RESULT PrlVmCfg_SetCapabilitiesMask(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nCapMask
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nCapMask

[in] A bitmask containing the supported capabilities. For available options see the PRL_CT_CAPMASK (p. 1046) enumeration and CAPDEFAULTMASK (p. 1661) defined in PrlEnums.h as macros.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetCapabilitiesMask (p. 559)

PrlVmCfg_SetCloseAppOnShutdown Function

Allows to specify whether the Virtuozzo application (such Virtuozzo Desktop) will be closed on the virtual machine shutdown.

Syntax

```
PRL_RESULT PrlVmCfg_SetCloseAppOnShutdown(
```

```
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmCloseAppOnShutdown
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

bVmCloseAppOnShutdown

Action type. Specify PRL_TRUE (p. 1652) to close the application. Specify PRL_FALSE (p. 1626) to keep it running.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsCloseAppOnShutdown (p. 621)

PrlVmCfg_SetConfirmationsList Function

Determines a list of operations with virtual machine which required administrator confirmation

Syntax

```
PRL_RESULT PrlVmCfg_SetConfirmationsList(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hConfirmList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hConfirmList

A handle of type is PHT_OPAQUE_TYPE_LIST (p. 177) contains new list of operations(values in enum PRL_ALLOWED_VM_COMMAND (p. 1036)) that required administrator's confirmation.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed, or device index out of bound

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a handle.

See Also

PrlSrv_DisableConfirmationMode (p. 236)() PrlSrv_EnableConfirmationMode (p. 236)() PrlSrv_IsConfirmationModeEnabled (p. 268)() PrlVmCfg_GetConfirmationsList (p. 561)() PrlVmCfg_SetConfirmationsList() PrlDispCfg_GetConfirmationsList (p. 39)() PrlDispCfg_SetConfirmationsList (p. 60)()

PrlVmCfg_SetCpuAccelLevel Function

Sets CPU acceleration level for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuAccelLevel(
    PRL_HANDLE hVmCfg,
    PRL_VM_ACCELERATION_LEVEL nVmCpuAccelLevel
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmCpuAccelLevel

The CPU acceleration level to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetCpuCount Function

Sets the number of CPUs for the specified virtual machine (the CPUs should be present in the machine).

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmCpuCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmCpuCount

The number to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetCpuHotplugEnabled Function

Sets CPU hotplug mode (enabled or disabled) for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuHotplugEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmCpuHotplugEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmCpuHotplugEnabled

The CPU hotplug mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetCpuLimit Function

Sets the CPU usage limit (in percent) for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuLimit(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmCpuLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmCpuLimit

CPU usage limit from the range of 0-100%.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetCpuLimit (p. 562)

PrlVmCfg_SetCpuLimitEx Function

Sets limit of CPU usage for the virtual machine in Mhz.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuLimitEx(
    PRL_HANDLE hVmCfg,
    PRL_CONST_CPULIMIT_DATA_PTR pVmCpuLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. cpulimit A cpulimit data of PRL_CPULIMIT_DATA type

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetCpuMask Function

Sets the CPU affinity for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuMask(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sMask
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sMask

CPU affinity mask is represented by an cpu number or cpu range separated by the comma. (0,2,3-10)

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetCpuLimit (p. 562)

PrlVmCfg_SetCpuMode Function

Sets CPU mode (32 bit or 64 bit) for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuMode(
    PRL_HANDLE hVmCfg,
    PRL_CPU_MODE nVmCpuMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmCpuMode

The CPU mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetCpuUnits Function

Sets the number of CPU units that will be allocated to a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetCpuUnits(
    PRL_HANDLE hVmCfg,
```

```
    PRL_UINT32 nVmCpuUnits
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmCpuUnits

The number of CPU units to allocate.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Each virtual machine on the same host can be allocated a number of CPU units from the range of 8 to 500000. The larger the number is, the more CPU time the virtual machine will get.

See Also

PrlVmCfg_GetCpuUnits (p. 565)

PrlVmCfg_SetCustomProperty Function

Sets the virtual machine custom property information.

Syntax

```
PRL_RESULT PrlVmCfg_SetCustomProperty(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmCustomProperty
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmCustomProperty

A pointer to a buffer containing the custom information. The data must be in CDATA format.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If you have some custom data, which relates to a virtual machine in some way, you can save it in the virtual machine configuration for future retrieval. This can be any data that your client application requires. The data is stored in CDATA format.

PrlVmCfg_SetDefaultConfig Function

The PrlVmCfg_SetDefaultConfig function selects a default configuration for a new virtual machine based on the operating system that the virtual machine will be running.

Syntax

```
PRL_RESULT PrlVmCfg_SetDefaultConfig(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hSrvConfig,
    PRL_UINT32 guestOsVersion,
    PRL_BOOL needCreateDevices
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

guestOsVersion

The version of the operating system that the virtual machine will be running. The OS versions can be found in the PrlOses.h file. The names of the macros are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

needCreateDevices

A flag indicating whether default virtual machine devices should be created or not. Specify PRL_TRUE (p. 1652) to create default devices, or PRL_FALSE (p. 1626) to skip the device creation.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The default configuration information is obtained from the host machine configuration, which is passed to the function as a parameter. Using a default configuration for a new virtual machine is the simplest and safest way to configure a new virtual machine. For the example on how to create a new virtual machine using a default configuration, see the PrlVm_Reg (p. 483) function.

PrlVmCfg_SetDescription Function

Sets the virtual machine description.

Syntax

```
PRL_RESULT PrlVmCfg_SetDescription(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmDescription
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmDescription

The virtual machine description to set. The value must be a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetDisableAPICSign Function

Allows to specify whether the virtual machine should be using APIC during runtime.

Syntax

```
PRL_RESULT PrlVmCfg_SetDisableAPICSign(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bDisableAPIC
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bDisableAPIC

The value to set. Specify PRL_TRUE (p. 1652) to enable the APIC. Specify PRL_FALSE (p. 1626) disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsDisableAPIC (p. 624)

PrlVmCfg_SetDisableSpeakerSign Function

Allows to specify whether the virtual machine should be using speaker during runtime.

Syntax

```
PRL_RESULT PrlVmCfg_SetDisableSpeakerSign(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bDisableSpeaker
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bDisableSpeaker

The value to set. Specify PRL_TRUE (p. 1652) to enable the speaker. Specify PRL_FALSE (p. 1626) disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsDisableSpeaker (p. 625)

PrlVmCfg_SetDiskCacheWriteBack Function

Sets the virtual machine disk cache write-back option.

Syntax

```
PRL_RESULT PrlVmCfg_SetDiskCacheWriteBack(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmDiskCacheWriteBack
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmDiskCacheWriteBack

The disk cache write-back option to set. Specify PRL_TRUE (p. 1652) to enable the disk cache write-back. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetDnsServers Function

Allows to manually assign DNS servers a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetDnsServers(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hDnsServersList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hDnsServersList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of DNS servers to be assigned to the network adapter. Can be PRL_INVALID_HANDLE (p. 1627) if it's necessary to specify an empty list.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network adapter specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in the target virtual machine, and its network adapter settings should be configured to change automatically.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

PrlVmCfg_SetEfiEnabled Function

Sets EFI boot for the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetEfiEnabled(
```

```
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEfiEnabled
);
```

File

PrlApiVm.h

Parameters

bEfiEnabled

Setting value. Perform PRL_TRUE (p. 1652) to switch on EFI boot. Perform PRL_FALSE (p. 1626) to switch off last one.

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid PHT_VIRTUAL_MACHINE (p. 442) handle or NULL pointer to the output buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

Remarks

for the specified virtual machine.

PrlVmCfg_SetExternalBootDevice Function

Sets an external boot device system name.

Syntax

```
PRL_RESULT PrlVmCfg_SetExternalBootDevice(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewSysName
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewSysName

The name to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetFeaturesMask Function

Sets a features inside the CT.

Syntax

```
PRL_RESULT PrlVmCfg_SetFeaturesMask(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nOn,
    PRL_UINT32 nOff
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nOn

[in] A bitmask of enabled features. For available options see the PRL_CT_FEATURES (p. 1047) enumeration

nOff

[in] A bitmask of disabled features. For available options see the PRL_CT_FEATURES (p. 1047) enumeration

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetCapabilitiesMask (p. 559)

PrlVmCfg_SetForegroundPriority Function

Sets the virtual machine foreground processes priority.

Syntax

```
PRL_RESULT PrlVmCfg_SetForegroundPriority(
    PRL_HANDLE hVmCfg,
    PRL_VM_PRIORITY nVmForegroundPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmForegroundPriority

The priority level to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetFreeDiskSpaceRatio Function

Sets the time free disk space ratio at which compacting virtual disks are doing by Automatic HDD compress.

Syntax

```
PRL_RESULT PrlVmCfg_SetFreeDiskSpaceRatio(
    PRL_HANDLE hVmCfg,
    PRL_DOUBLE dFreeDiskSpaceRatio
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

dFreeDiskSpaceRatio

The space ratio to set, in percents.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Automatic HDD compress is feature that allows to make automatic compress of a virtual machine disks at predefined space ratio. This function allows to set the space ratio at which the automatic compress are taken.

See Also

PrlVmCfg_GetFreeDiskSpaceRatio (p. 576)

PrlVmCfg_SetGuestSharingAutoMount Function

Sets the guest OS sharing auto-mount option.

Syntax

```
PRL_RESULT PrlVmCfg_SetGuestSharingAutoMount(
    PRL_HANDLE hVmCfg,
```

```
    PRL_BOOL bVmGuestSharingAutoMount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmGuestSharingAutoMount

Auto-mount option to set. Specify PRL_TRUE (p. 1652) to enable sharing auto-mount. Specify PRL_FALSE (p. 1626) to disable auto-mount.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If guest sharing is enabled (see PrlVmCfg_SetGuestSharingEnabled (p. 673)) and auto-mount is turned on, the guest OS disk drives will appear as icons on the host desktop. If sharing is enabled but auto-mount is not, you will not see the drives in the host OS but Smart Select feature will still work.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetGuestSharingEnabled Function

Enables the guest sharing feature.

Syntax

```
PRL_RESULT PrlVmCfg_SetGuestSharingEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmGuestSharingEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmGuestSharingEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable guest sharing. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If guest sharing is enabled, the guest OS disk drives will be accessible in the host OS. If, in addition, the auto-mount feature is turned on, the guest OS disk drives will appear as icons on the host desktop. If sharing is enabled but auto-mount is not, you will not see the drives in the host OS but Smart Select feature will still work.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetGuestSharingEnableSpotlight Function

Sets the VM virtual disks added to Spotlight search subsystem sign.

Syntax

```
PRL_RESULT PrlVmCfg_SetGuestSharingEnableSpotlight(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmGuestSharingEnableSpotlight
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmGuestSharingEnableSpotlight

Enable Spotlight option to set. Specify PRL_TRUE (p. 1652) to enable using virtual harddisks at Spotlight search subsystem. Specify PRL_FALSE (p. 1626) to disable virtual hard disks from Spotlight.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

If guest sharing is enabled (see PrlVmCfg_SetGuestSharingEnabled (p. 673)) and Spotlight enable sign is turned on, the search through Spotlight subsystem will be using VM virtual hardisks too.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetHighAvailabilityEnabled Function

Enables or disables the High Availability feature for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetHighAvailabilityEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
```

```
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable the High Availability feature. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsHighAvailabilityEnabled (p. 628)

PrlVmCfg_SetHighAvailabilityPriority Function

Sets the priority of the virtual machine in the High Availability Cluster.

Syntax

```
PRL_RESULT PrlVmCfg_SetHighAvailabilityPriority(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nPriority

The priority value to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetHighAvailabilityPriority (p. 581)

PrlVmCfg_SetHostname Function

Sets the specified virtual machine hostname.

Syntax

```
PRL_RESULT PrlVmCfg_SetHostname(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmHostname
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmName

The hostname to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetHostSharingEnabled Function

Enables host sharing for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetHostSharingEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmHostSharingEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmHostSharingEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable sharing. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When host sharing is enabled, the host OS shared folders will be visible in the guest OS. To use the host sharing feature, the shared folders must be set up first using the functionality provided by the PHT_SHARE (p. 350) object.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetIcon Function

Sets the virtual machine icon.

Syntax

```
PRL_RESULT PrlVmCfg_SetIcon(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmIcon
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmIcon

The new icon file name.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetIoLimit Function

Sets limit of I/O usage for the virtual machine in Bytes/s.

Syntax

```
PRL_RESULT PrlVmCfg_SetIoLimit(
    PRL_HANDLE hVmCfg,
    PRL_CONST_IOLIMIT_DATA_PTR pVmIoLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. pVmIoLimit A I/O limit data of

PRL_IOLIMIT_DATA type

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetIoPriority Function

Sets the specified virtual machine I/O priority.

Syntax

```
PRL_RESULT PrlVmCfg_SetIoPriority(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmIoPriority
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmIoPriority

io priority 0-7.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Priority range is [PRL_IOPRIO_MIN - PRL_IOPRIO_MAX]. The greater priority is, the more time for I/O activity Vm has. By default each Vm has priority of IO_PRIO_DEFAULT.

PrlVmCfg_SetIopsLimit Function

Sets the specified virtual machine IOPS limit.

Syntax

```
PRL_RESULT PrlVmCfg_SetIopsLimit(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmIopsLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. nVmIopsLimit IOPS limit.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetIsolatedVmEnabled Function

Disables or enables shared camera isolated VM.

Syntax

```
PRL_RESULT PrlVmCfg_SetIsolatedVmEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetLockGuestOnSuspendEnabled Function

Disables or enables shared camera lock guest on suspend.

Syntax

```
PRL_RESULT PrlVmCfg_SetLockGuestOnSuspendEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetLockInFullScreenMode Function

Enables or disables the lock in screen mode feature in a virtual machine configuration.

Syntax

```
PRL_RESULT PrlVmCfg_SetLockInFullScreenMode(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bValue
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bValue

Action type. Specify PRL_TRUE (p. 1652) to enable feature. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Lock in screen mode is feature of Virtuozzo Player product that allows to mandatory apply full screen mode for Virtuozzo GUI until guest OS started and go to the windowed mode just after guest OS will be stopped.

See Also

PrlVmCfg_IsLockInFullScreenMode (p. 631)

PrlVmCfg_SetLongerBatteryLifeEnabled Function

Disables or enables longer battery life.

Syntax

```
PRL_RESULT PrlVmCfg_SetLongerBatteryLifeEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetMapSharedFoldersOnLetters Function

Enables mapping of shared host disks on drive letters for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetMapSharedFoldersOnLetters(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bMapSharedFoldersOnLetters
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bMapSharedFoldersOnLetters

The value to set. Specify PRL_TRUE (p. 1652) to map shared host disks on drive letters. Specify PRL_FALSE (p. 1626) to not share it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When this sign switched on host disks shared into guest Windows OS will be maped on correspond drive letters. Note that host sharing should be switched on to enable this functionality.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetMemGuaranteeSize Function

Sets the minimum memory limit which is guaranteed for the virtual machine, in percent of guest RAM.

Syntax

```
PRL_RESULT PrlVmCfg_SetMemGuaranteeSize(
    PRL_HANDLE hVmCfg,
    PRL_CONST_MEMGUARANTEE_DATA_PTR pMemGuaranteeSize
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

pMemGuaranteeSize

The memory size structure to set, value and type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetName Function

Sets the specified virtual machine name.

Syntax

```
PRL_RESULT PrlVmCfg_SetName(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmName
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmName

The name to set. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetNestedVirtualizationEnabled Function

Disables or enables nested virtualization.

Syntax

```
PRL_RESULT PrlVmCfg_SetNestedVirtualizationEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetNetfilterMode Function

Allows to set netfilter mode inside the CT.

Syntax

```
PRL_RESULT PrlVmCfg_SetNetfilterMode(
    PRL_HANDLE hVmCfg,
    PRL_NETFILTER_MODE nMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nMode

[in] The netfilter mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetNetfilterMode (p. 590)

PrlVmCfg_SetNetworkRateList Function

Add the specified rates list to the Vm configuration.

Syntax

```
PRL_RESULT PrlVmCfg_SetNetworkRateList(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hNetworkRateList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

phNetworkRateList

A handle of type PHT_HANDLES_LIST (p. 108) contains the list of handle sof type PHT_NETWORK_RATE (p. 157).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlSrv_GetNetworkShapingConfig (p. 251)

PrlVmCfg_SetNodeMask Function

Sets the NUMA node mask for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetNodeMask(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sMask
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sMask

NUMA node mask represented by a CPU number or CPU range separated by comma, e.g., 0,2,3-10.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetCpuLimit (p. 562)

PrlVmCfg_SetOfflineManagementEnabled Function

Enables or disables the offline management feature for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetOfflineManagementEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable the offline management feature. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetOfflineServices Function

Allows to manually set offline services that will be available in the virtual machine offline management.

Syntax

```
PRL_RESULT PrlVmCfg_SetOfflineServices(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hOfflineServicesList
);
```

File

PrlApiVm.h

Parameters

hOfflineServicesList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of offline services. Specify PRL_INVALID_HANDLE (p. 1627) to assign the default list of services.

hVmDev

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual machine specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_GetOfflineServices (p. 592)

PrlVmCfg_SetOptimizeModifiersMode Function

Sets optimize modifiers mode.

Syntax

```
PRL_RESULT PrlVmCfg_SetOptimizeModifiersMode(
    PRL_HANDLE hVm,
    PRL_OPTIMIZE_MODIFIERS_MODE nMode
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nMode

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetOptionsOnGuestCrash Function

%VM_ONLY%

Allows to set the on-crash options for the specified virtual machine

Syntax

```
PRL_RESULT PrlVmCfg_SetOptionsOnGuestCrash(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nOptions
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nOptions

Additional options to set (PRL_VM_ON_CRASH_OPTION (p. 948))

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetOsResInFullScrMode Function

Sets the virtual machine OS resolution in full screen mode option.

Syntax

```
PRL_RESULT PrlVmCfg_SetOsResInFullScrMode(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmOsResInFullScrMode
```

```
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmOsResInFullScrMode

The value to set. Specify PRL_TRUE (p. 1652) to enable the option. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetOsTemplate Function

Specify the Virtual Machine Operation System template name which will be used to create new Virtuozzo Virtuozzo Container.

Syntax

```
PRL_RESULT PrlVmCfg_SetOsTemplate(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sOsTemplate
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. sOsTemplate ostemplate name.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

This option acceptable for Virtual Machine of PVT_CT only.

PrlVmCfg_SetOsVersion Function

Sets the virtual machine guest OS version.

Syntax

```
PRL_RESULT PrlVmCfg_SetOsVersion(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmOsVersion
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmOsVersion

The guest OS version to set. The OS versions are defined in the PrlOses.h file as macros. Look for the macros that are prefixed with PVS_GUEST_VER_ (e.g. PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetPMUVirtualizationEnabled Function

Disables or enables PMU virtualization.

Syntax

```
PRL_RESULT PrlVmCfg_SetPMUVirtualizationEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetProfile Function

Set one of predefined VM profiles.

Syntax

```
PRL_RESULT PrlVmCfg_SetProfile(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hSrvConfig,
    PRL_VIRTUAL_MACHINE_PROFILE nVmProfile
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hSrvConfig

A handle of type PHT_SERVER_CONFIG (p. 314).

nVmProfile

[in] A predefined settings profile. See the PRL_VIRTUAL_MACHINE_PROFILE (p. 1100) enumeration for available options.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetRamHotplugEnabled Function

Sets the virtual machine memory (RAM) hotplug mode (enabled or disabled).

Syntax

```
PRL_RESULT PrlVmCfg_SetRamHotplugEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmRamHotplugEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmRamHotplugEnabled

The RAM hotplug mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetRamSize Function

Sets the virtual machine memory (RAM) size.

Syntax

```
PRL_RESULT PrlVmCfg_SetRamSize(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmRamSize
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmRamSize

The RAM size to set, in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetRateBound Function

Sets the ratebound flag.

Syntax

```
PRL_RESULT PrlVmCfg_SetRateBound(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

If this parameter is set to PRL_TRUE (p. 1652) then Vm output bandwidth specified by rate will be bandwidth limit and guarantee. Otherwise rate sets Vm output bandwidth guarantee for traffic and limit is set by TOTALRATE in the network shaping Configuratio,.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetResource Function

Sets barrier and limit for various user beancounters.

Syntax

```
PRL_RESULT PrlVmCfg_SetResource(
    PRL_HANDLE hVmCfg,
    PRL_CT_RESOURCE nResourceId,
    PRL_UINT64 nBarrirer,
    PRL_UINT64 nLimit
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. nResourceIdThe resource id (Only PCR_SWAPPAGES supported for now). nBarrirerThe resource barrier. nLimit The resource limit

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetScrResEnabled Function

Enables or disables the additional screen resolution support in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetScrResEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmScrResEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmScrResEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable the additional screen resolution support. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsScrResEnabled (p. 636)

PrlVmCfg_SetSearchDomains Function

Allows to manually set the global search domain list that will be assigned to the Virtuozzo Container.

Syntax

```
PRL_RESULT PrlVmCfg_SetSearchDomains(
    PRL_HANDLE hVmCfg,
    PRL_HANDLE hSearchDomainsList
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

hSearchDomainsList

A handle of type PHT_STRINGS_LIST (p. 357) containing the list of search domains to be applied to the global network settings of the virtual machine operating system. Can be PRL_INVALID_HANDLE (p. 1627) if it's necessary to specify an empty list.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle of virtual network adapter specified.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Note: For these changes to take effect, Virtuozzo Tools must be installed in a target virtual machine.

If the changes are applied to a stopped virtual machine, they will take effect the next time the machine is started. If the changes are applied to a running virtual machine, they will take effect on configuration changes commit.

See Also

PrlVmCfg_GetSearchDomains (p. 600)

PrlVmCfg_SetShareAllHostDisks Function

Enables sharing of all host disks for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetShareAllHostDisks(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bShareAllHostDisks
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bShareAllHostDisks

The value to set. Specify PRL_TRUE (p. 1652) to share all host disks to guest OS. Specify PRL_FALSE (p. 1626) to not share them.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When this sign switched on all host disks will be shared into guest OS. Note that host sharing should be switched on to enable this functionality.

Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetShareClipboard Function

Enables or disables the clipboard sharing feature.

Syntax

```
PRL_RESULT PrlVmCfg_SetShareClipboard(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Set to PRL_TRUE (p. 1652) to enabled clipboard sharing. Set to PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When clipboard sharing is enabled, you can copy and paste text and files from the host to a virtual machine or vise versa, or from one virtual machine to another. This feature requires Virtuozzo Tools installed in a virtual machine.

PrlVmCfg_SetSharedProfileEnabled Function

Enables or disables the Shared Profile feature in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetSharedProfileEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable the Shared Profile. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsSharedProfileEnabled (p. 637)

PrlVmCfg_SetShareUserHomeDir Function

Enables or disables sharing of the host user home directory in the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetShareUserHomeDir(
    PRL_HANDLE hVmCfg,
```

```
    PRL_BOOL bShareUserHomeDir
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bShareUserHomeDir

The value to set. Specify PRL_TRUE (p. 1652) to share the host user home directory. Specify PRL_FALSE (p. 1626) not to share it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

When this feature is enabled, the host user home directory will be shared in the virtual machine. Note that host sharing should be enabled to use this functionality. Virtuozzo Tools must be installed to use this feature. To obtain the Virtuozzo Tools status, use the PrlVm_GetToolsState (p. 471) function.

PrlVmCfg_SetSmartGuardEnabled Function

Enables the SmartGuard feature in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartGuardEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable SmartGuard. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is feature that allows to make automatic snapshots of a virtual machine at predefined time intervals. The feature

requires Virtuozzo Tools installed in a virtual machine.

See Also

PrlVmCfg_SetSmartGuardInterval (p. 697)

PrlVmCfg_SetSmartGuardMaxSnapshotsCount (p. 697)

PrlVmCfg_SetSmartGuardNotifyBeforeCreation (p. 698)

PrlVmCfg_SetSmartGuardInterval Function

Sets the time interval at which snapshots are taken by SmartGuard.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartGuardInterval(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nInterval
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nInterval

The time interval to set, in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is feature that allows to make automatic snapshots of a virtual machine at predefined time intervals. This function allows to set the interval at which the automatic snapshots are taken.

See Also

PrlVmCfg_GetSmartGuardInterval (p. 604)

PrlVmCfg_SetSmartGuardMaxSnapshotsCount Function

Sets the maximum snapshot count, a SmartGuard feature.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartGuardMaxSnapshotsCount(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nMaxSnapshotsCount
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nMaxSnapshotsCount

Maximum snapshot count to set. The maximum allowable value is 100.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is a facility that allows to make automatic snapshots of a virtual machine at predefined time intervals. The snapshot count setting specifies how many snapshots taken by SmartGuard will be kept on the hard drive (maximum is 100). As soon as the maximum number is exceeded, the oldest snapshot is deleted.

PrlVmCfg_SetSmartGuardNotifyBeforeCreation Function

Enables or disables notification of automatic snapshot creation, a SmartGuard feature.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartGuardNotifyBeforeCreation(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bNotifyBeforeCreation
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bNotifyBeforeCreation

Action type. Specify PRL_TRUE (p. 1652) if you want the user to be notified. Specify PRL_FALSE (p. 1626) if you want to create snapshots silently.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

SmartGuard is a facility that allows to make automatic snapshots of a virtual machine at predefined time intervals. By default, the snapshots are creating transparently to the user. If needed, Virtuozzo Service can notify the user that a snapshot is about to

be created. The user can then choose to proceed with snapshot creation or cancel it. This function allows to enabled or disable this feature. The notifications are received by the client via the PHT_EVENT (p. 72) handle received by a callback function. See PHT_EVENT (p. 72) and PrlEvent_CreateAnswerEvent (p. 74) for more information and code samples.

See Also

PrlVmCfg_IsSmartGuardNotifyBeforeCreation (p. 639)

PrlVmCfg_SetSmartMountDVDsEnabled Function

Enables or disables the SmartMount CD/DVD drives.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartMountDVDsEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable shared folder. Specify PRL_FALSE (p. 1626) to disable them.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetSmartMountEnabled Function

Enables or disables the SmartMount feature.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartMountEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable shared folder. Specify PRL_FALSE (p. 1626) to disable them.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetSmartMountNetworkSharesEnabled Function

Enables or disables the SmartMount network shares.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartMountNetworkSharesEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable shared folder. Specify PRL_FALSE (p. 1626) to disable them.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetSmartMountRemovableDrivesEnabled Function

Enables or disables the SmartMount removable drives.

Syntax

```
PRL_RESULT PrlVmCfg_SetSmartMountRemovableDrivesEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable shared folder. Specify PRL_FALSE (p. 1626) to disable them.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetStartInDetachedWindowEnabled Function

Disables or enables start in detached window.

Syntax

```
PRL_RESULT PrlVmCfg_SetStartInDetachedWindowEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetStartLoginMode Function

Allows to set the automatic startup login mode for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetStartLoginMode(
    PRL_HANDLE hVmCfg,
    PRL_VM_START_LOGIN_MODE nVmStartLoginMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmStartLoginMode

Automatic startup login mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetStartUserCreds Function

Allows to set the automatic startup user login and password for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetStartUserCreds(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sStartUserLogin,
    PRL_CONST_STR sPassword
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sStartUserLogin

User login. Must be a a UTF-8 encoded, null-terminated string.

sPassword

User password. Must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetSwitchOffWindowsLogoEnabled Function

Disables or enables Windows logo.

Syntax

```
PRL_RESULT PrlVmCfg_SetSwitchOffWindowsLogoEnabled(
    PRL_HANDLE hVm,
    PRL_BOOL bEnable
);
```

File

PrlApiVm.h

Parameters

hVm

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnable

[in] Setting value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetSystemFlags Function

Sets the virtual machine system flags.

Syntax

```
PRL_RESULT PrlVmCfg_SetSystemFlags(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmSystemFlags
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VIRTUAL_MACHINE (p. 442) identifying the virtual machine.

unnamed_param

New virtual machine system flags, a UTF-8 encoded null terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetTemplateSign Function

Allows to modify a regular virtual machine to become a template, and vise versa.

Syntax

```
PRL_RESULT PrlVmCfg_SetTemplateSign(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bVmIsTemplate
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bVmIsTemplate

The value to set. Specify PRL_TRUE (p. 1652) to make the virtual machine a template. Specify PRL_FALSE (p. 1626) to switch a template to a regular virtual machine.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A template is very similar to a regular virtual machine with the major exception being it cannot be started. Templates are used to create new virtual machines as exact copies of them. To create a virtual machine from a template, use the PrlVm_Clone (p. 450) function specifying the template as the source.

PrlVmCfg_SetTimeSynchronizationEnabled Function

Enables or disables the time synchronization feature in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetTimeSynchronizationEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

You can synchronize the time in a virtual machine with the time in the host OS. To do so, the time synchronization feature must be enabled using the PrlVmCfg_SetTimeSynchronizationEnabled function. The time will be synchronized at predefined time intervals. To set the time interval, use the PrlVmCfg_SetSmartGuardInterval (p. 697) function. This feature requires Virtuozzo Tools installed in a virtual machine.

See Also

PrlVmCfg_SetTimeSyncSmartModeEnabled (p. 706)

PrlVmCfg_SetTimeSyncInterval Function

Allows to set the time interval at which the time in the virtual machine will be synchronized with the time in the host OS.

Syntax

```
PRL_RESULT PrlVmCfg_SetTimeSyncInterval(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nTimeSyncInterval
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nTimeSyncInterval

Setting interval in seconds.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Before this function can be used, the time synchronization feature must be enabled in the virtual machine using the PrlVmCfg_SetTimeSynchronizationEnabled (p. 704) function.

See Also

PrlVmCfg_GetTimeSyncInterval (p. 609)

PrlVmCfg_SetTimeSyncSmartModeEnabled Function

Enables or disables the smart time-synchronization mode in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetTimeSyncSmartModeEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

You can automatically synchronize the time in a virtual machine with the time in the OS by using the PrlVmCfg_SetTimeSynchronizationEnabled (p. 704) function. That functionality will keep the time in the virtual machine in a perfect synch with the time in the host OS. You can also set different time zones in the host and the guest OS and still synchronize the time. To do so, use the PrlVmCfg_SetTimeSyncSmartModeEnabled function. This feature requires Virtuozzo Tools installed in a virtual machine.

See Also

PrlVmCfg_SetTimeSynchronizationEnabled (p. 704)

PrlVmCfg_SetTimeSyncInterval (p. 705)

PrlVmCfg_SetToolsAutoUpdateEnabled Function

Enables or disables the Virtuozzo Tools AutoUpdate feature for a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetToolsAutoUpdateEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable the AutoUpdate feature. Specify PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUndoDisksMode Function

Sets the undo-disks mode for the virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetUndoDisksMode(
    PRL_HANDLE hVmCfg,
    PRL_UNDO_DISKS_MODE nUndoDisksMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nUndoDisksMode

The undo disks mode option to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUseDefaultAnswers Function

Enables the use default answers mechanism in a virtual machine.

Syntax

```
PRL_RESULT PrlVmCfg_SetUseDefaultAnswers(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bUseDefaultAnswers
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Action type. Specify PRL_TRUE (p. 1652) to enable mechanism. In this case system won't passing questions from virtual machine to client side and answering on them automatically. Specify PRL_FALSE (p. 1626) to disable mechanism and processing virtual machine questions explicitly.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmCfg_IsUseDefaultAnswers (p. 646)

PrlVmCfg_SetUseDesktop Function

Enables or disables the "undo-desktop" feature in the shared profile.

Syntax

```
PRL_RESULT PrlVmCfg_SetUseDesktop(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUseDocuments Function

Lets to set use documents in shared profile enabling sign.

Syntax

```
PRL_RESULT PrlVmCfg_SetUseDocuments(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUseDownloads Function

Lets to set use downloads in shared profile enabling sign.

Syntax

```
PRL_RESULT PrlVmCfg_SetUseDownloads(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUseMovies Function

Lets to set use movies in shared profile enabling sign.

Syntax

```
PRL_RESULT PrlVmCfg_SetUseMovies(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUseMusic Function

Lets to set use music in shared profile enabling sign.

Syntax

```
PRL_RESULT PrlVmCfg_SetUseMusic(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUsePictures Function

Lets to set use pictures in shared profile enabling sign.

Syntax

```
PRL_RESULT PrlVmCfg_SetUsePictures(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

Setting boolean value.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUserDefinedSharedFoldersEnabled Function

Enables or disables user-defined shared folders.

Syntax

```
PRL_RESULT PrlVmCfg_SetUserDefinedSharedFoldersEnabled(
    PRL_HANDLE hVmCfg,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

bEnabled

The value to set. Specify PRL_TRUE (p. 1652) to enable shared folder. Specify PRL_FALSE (p. 1626) to disable them.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetUuid Function

Sets the virtual machine UUID (universally unique ID).

Syntax

```
PRL_RESULT PrlVmCfg_SetUuid(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmUuid
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmUuid

The UUID to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

A UUID is assigned to a virtual machine automatically when it is created. Before using this function, make sure that you understand what UUID is, and that you have a good reason for changing it.

PrlVmCfg_SetVideoRamSize Function

Sets the virtual machine video memory size.

Syntax

```
PRL_RESULT PrlVmCfg_SetVideoRamSize(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmVideoRamSize
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmVideoRamSize

The new memory size, in megabytes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetVmType Function

Specify the Virtual Machine type on create action.

Syntax

```
PRL_RESULT PrlVmCfg_SetVmType(
    PRL_HANDLE hVmCfg,
    PRL_VM_TYPE nType
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration. nType Virtuam Machine type.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetVNCHostName Function

Sets the virtual machine VNC host name.

Syntax

```
PRL_RESULT PrlVmCfg_SetVNCHostName(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmRemoteDisplayHostName
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmRemoteDisplayHostName

The new VNC host name. The value must be a a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetVNCMode Function

Sets the virtual machine VNC mode.

Syntax

```
PRL_RESULT PrlVmCfg_SetVNCMode(
    PRL_HANDLE hVmCfg,
    PRL_VM_REMOTE_DISPLAY_MODE nVmRemoteDisplayMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmRemoteDisplayMode

The VNC mode to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetVNCPassword Function

Sets the virtual machine VNC password.

Syntax

```
PRL_RESULT PrlVmCfg_SetVNCPassword(
    PRL_HANDLE hVmCfg,
    PRL_CONST_STR sNewVmRemoteDisplayPassword
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

sNewVmRemoteDisplayPassword

The VNC password to set. The value must be a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetVNCPort Function

Sets the virtual machine VNC port number.

Syntax

```
PRL_RESULT PrlVmCfg_SetVNCPort(
    PRL_HANDLE hVmCfg,
    PRL_UINT32 nVmRemoteDisplayPort
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmRemoteDisplayPort

The port number to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmCfg_SetWindowMode Function

Sets the virtual machine window mode.

Syntax

```
PRL_RESULT PrlVmCfg_SetWindowMode(
    PRL_HANDLE hVmCfg,
```

```
    PRL_VM_WINDOW_MODE nVmWindowMode
);
```

File

PrlApiVm.h

Parameters

hVmCfg

A handle of type PHT_VM_CONFIGURATION (p. 537) identifying the virtual machine configuration.

nVmWindowMode

The window mode to set. See the PRL_VM_WINDOW_MODE (p. 952) enumeration for the list of modes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PHT_VM_GUEST_SESSION

The PHT_VM_GUEST_SESSION handle is used to run programs and execute administrative tasks in the virtual machine. The handle can also be used to obtain some of the guest operating system configuration information. The handle is obtained using the PrlVm_LoginInGuest (p. 473) function.

Functions

	Name	Description
	PrlVmGuest_GetNetworkSettings (p. 716)	Obtains network settings of a guest operating system running in a virtual machine.
	PrlVmGuest_Logout (p. 717)	Closes a session (or unbinds from a pre-existing session) in a virtual machine that was previously opened with the PrlVm_LoginInGuest (p. 473) function.
	PrlVmGuest_RunProgram (p. 718)	Executes a program in a virtual machine.

PrlVmGuest_GetNetworkSettings Function

Obtains network settings of a guest operating system running in a virtual machine.

Syntax

```
PRL_HANDLE PrlVmGuest_GetNetworkSettings(
    PRL_HANDLE hVmGuest,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

hVmGuest

A handle of type PHT_VM_GUEST_SESSION (p. 716) identifying the user session in a virtual machine. The handle is obtained using the PrlVm_LoginInGuest (p. 473) function.

nReserved

Reserved parameter for future purposes. Must be 0 in the current implementation.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

The network settings are obtained using a user session in the guest operating system. Prior to calling this function, a handle to a user session in the virtual machine must be obtained using the PrlVm_LoginInGuest (p. 473) function.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

To get the results from the returned PHT_JOB (p. 138) object, use the following steps:

1. Use the PrlJob_GetResult (p. 142) function to obtain a handle of type PHT_RESULT (p. 210).
2. Use the PrlResult_GetParam (p. 210) function to obtain a handle of type PHT_SERVER_CONFIG (p. 314). The handle will contain the guest OS network settings.

Once you obtain the PHT_SERVER_CONFIG (p. 314) handle, you can use the following functions to obtain individual network settings of the guest OS:

PrlSrvCfg_GetDnsServers (p. 322)

PrlSrvCfg_GetHostname (p. 327)

PrlSrvCfg_GetSearchDomains (p. 337)

PrlSrvCfg_GetNetAdapter (p. 332)

PrlSrvCfg_GetDefaultGateway (p. 320)

PrlVmGuest_Logout Function

Closes a session (or unbinds from a pre-existing session) in a virtual machine that was previously opened with the PrlVm_LoginInGuest (p. 473) function.

Syntax

```
PRL_HANDLE PrlVmGuest_Logout(
    PRL_HANDLE hVmGuest,
    PRL_UINT32 nReserved
);
```

File

PrlApiVm.h

Parameters

hVmGuest

A handle of type PHT_VM_GUEST_SESSION (p. 716) identifying the session.

nReserved

Reserved parameter for future purposes. Must contain 0 in the current implementation.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_LoginInGuest (p. 473)

PrlVmGuest_RunProgram Function

Executes a program in a virtual machine.

Syntax

```
PRL_HANDLE PrlVmGuest_RunProgram(
    PRL_HANDLE hVmGuest,
    PRL_CONST_STR sAppName,
    PRL_HANDLE hArgsList,
    PRL_HANDLE hEnvsList,
    PRL_UINT32 nFlags,
    PRL_FILE_DESC nStdin,
    PRL_FILE_DESC nStdout,
    PRL_FILE_DESC nStderr
);
```

File

PrlApiVm.h

Parameters

hVmGuest

A handle of type PHT_VM_GUEST_SESSION (p. 716) identifying the user session in a virtual machine. The handle is obtained using the PrlVm_LoginInGuest (p. 473) function.

sAppName

An absolute path to the program that you would like to execute.

hArgsList

A handle of type PHT_STRINGS_LIST (p. 357) specifying the list of arguments to pass to the program. Each token must be a separate list item.

hEnvsList

A handle of type PHT_STRINGS_LIST (p. 357) specifying the list of environment variables to add to the program execution environment. Each variable must be a separate list item in the var_name=var_value format.

nFlags

Modificators of the program execution. Acceptable values:

PFD_STDOUT - return stdout.

PFD_STDERR - return stderr.

PFD_ALL - return both stdout and stderr (a blank nFlags parameter yields the same result).

PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY - run the program and return immediately without waiting for the program execution results.

nStdin

file descriptor on standard input (PRL_INVALID_FILE_DESCRIPTOR should be passed if you don't want to use file descriptors. Please note that IO channel between client and VM should be established before using file descriptors. Please call PrlDevDisplay_ConnectToVm in order to do it).

nStdout

file descriptor on standard output (PRL_INVALID_FILE_DESCRIPTOR should be passed if you don't want to use file descriptors. Please note that IO channel between client and VM should be established before using file descriptors. Please call PrlDevDisplay_ConnectToVm in order to do it).

nStderr

file descriptor on standard errors output (PRL_INVALID_FILE_DESCRIPTOR should be passed if you don't want to use file descriptors. Please note that IO channel between client and VM should be established before using file descriptors. Please call PrlDevDisplay_ConnectToVm in order to do it).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

Prior to calling this function, a handle to a user session in a virtual machine must be obtained using the PrlVm_LoginInGuest (p. 473) function. Once you obtain a session handle, you can use this function to run programs in the virtual machine. Depending on the options that were used during login, you will be able to run GUI applications and/or console programs. See the PrlVm_LoginInGuest (p. 473) function for more info.

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_VM_EXEC_PROGRAM_NOT_FOUND (p. 1394) - the specified program at the specified path was not found.

PRL_ERR_UNINITIALIZED (p. 1373) - file descriptors interface tried to use but PrlDevDisplay_ConnectToVm wasn't call.

PRL_ERR_SUCCESS (p. 1365) - program executed successfully.

PHT_VM_INFO

The PHT_VM_INFO handle is a container that contains the virtual machine state, access rights, and other information. The handle is obtained by making the PrlVm_GetState (p. 468) function call.

Functions

	Name	Description
	PrlVmInfo_GetAccessRights (p. 720)	Obtains a handle of type PHT_ACCESS_RIGHTS (p. 9).
	PrlVmInfo_GetAdditionState (p. 720)	Returns the virtual machine addition state information.
	PrlVmInfo_GetState (p. 721)	Returns the virtual machine state information.
	PrlVmInfo_IsInvalid (p. 721)	Determines if the specified virtual machine is invalid.
	PrlVmInfo_IsInvalidEx (p. 722)	Determines if the specified virtual machine is invalid.

	PrlVmInfo_IsVmWaitingForAnswer (p. 723)	Determines if the specified virtual machine is waiting for answer on self question.
	PrlVmInfo_IsVncServerStarted (p. 723)	Determines whether a VNC server is running for the specified virtual machine.

PrlVmInfo_GetAccessRights Function

Obtains a handle of type PHT_ACCESS_RIGHTS (p. 9).

Syntax

```
PRL_RESULT PrlVmInfo_GetAccessRights(
    PRL_HANDLE hVmInfo,
    PRL_HANDLE_PTR phVmAcl
);
```

File

PrlApiVm.h

Parameters

hVmInfo

A handle of type PHT_VM_INFO (p. 719).

phVmAcl

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create PHT_ACCESS_RIGHTS (p. 9) object.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine access rights information is not available.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmInfo_GetAdditionState Function

Returns the virtual machine addition state information.

Syntax

```
PRL_RESULT PrlVmInfo_GetAdditionState(
    PRL_HANDLE handle,
    VIRTUAL_MACHINE_ADDITION_STATE_PTR pVmState
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_INFO (p. 719).

pVmState

[out] A pointer to a variable that receives the virtual machine addition state information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine state information is not available.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmInfo_GetState Function

Returns the virtual machine state information.

Syntax

```
PRL_RESULT PrlVmInfo_GetState(
    PRL_HANDLE handle,
    VIRTUAL_MACHINE_STATE_PTR pVmState
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_INFO (p. 719).

pVmState

[out] A pointer to a variable that receives the virtual machine state information.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the virtual machine state information is not available.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmInfo_IsInvalid Function

Determines if the specified virtual machine is invalid.

Syntax

```
PRL_RESULT PrlVmInfo_IsInvalid(
    PRL_HANDLE handle,
    PRL_BOOL_PTR pbIsInvalid
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_INFO (p. 719).

pVmIsInvalid

[out] A pointer to a variable that receives result. PRL_TRUE (p. 1652) - the virtual machine is invalid. PRL_FALSE (p. 1626) - the virtual machine is fully operational.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the status of the virtual machine cannot be determined.

RL_ERR_SUCCESS - function completed successfully.

Remarks

A virtual machine can be reported as invalid by the function if its configuration has been corrupted, it's been removed from the virtual machine, and for other similar reasons.

PrlVmInfo_IsInvalidEx Function

Determines if the specified virtual machine is invalid.

Syntax

```
PRL_RESULT PrlVmInfo_IsInvalidEx(
    PRL_HANDLE handle,
    PRL_BOOL_PTR pbIsInvalid,
    PRL_RESULT_PTR pnInvalidReason
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_INFO (p. 719).

pVmIsInvalid

[out] A pointer to a variable that receives result. PRL_TRUE (p. 1652) - the virtual machine is invalid. PRL_FALSE (p. 1626) - the virtual machine is fully operational.

pVmInvalidReason

[out] A pointer to a variable that receives reason of invalidity as error code.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the status of the virtual machine cannot be determined.

RL_ERR_SUCCESS - function completed successfully.

Remarks

A virtual machine can be reported as invalid by the function if its configuration has been corrupted, it's been removed from the virtual machine, and for other similar reasons.

PrlVmInfo_IsVmWaitingForAnswer Function

Determines if the specified virtual machine is waiting for answer on self question.

Syntax

```
PRL_RESULT PrlVmInfo_IsVmWaitingForAnswer(
    PRL_HANDLE handle,
    PRL_BOOL_PTR pbIsVmWaitingForAnswer
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_INFO (p. 719).

pbIsVmWaitingForAnswer

[out] A pointer to a variable that receives result. PRL_TRUE (p. 1652) - the virtual machine is waiting for answer. PRL_FALSE (p. 1626) - the virtual machine is working normal.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the status of the virtual machine cannot be determined.

RL_ERR_SUCCESS - function completed successfully.

PrlVmInfo_IsVncServerStarted Function

Determines whether a VNC server is running for the specified virtual machine.

Syntax

```
PRL_RESULT PrlVmInfo_IsVncServerStarted(
    PRL_HANDLE handle,
    PRL_BOOL_PTR pbIsVncServerStarted
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_INFO (p. 719).

pbIsVncServerStarted

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that a VNC server is running. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the VNC server state information is not available.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVm_StartVncServer (p. 492)

PrlVm_StopVncServer (p. 494)

PHT_VM_TOOLS_INFO

The PHT_VM_TOOLS_INFO handle is used to determine whether the Virtuozzo Tools package is installed in the specified virtual machine and for obtaining its status and version information. The handle is obtained from the job object returned by the PrlVm_GetToolsState (p. 471) function.

Functions

	Name	Description
	PrlVmToolsInfo_GetState (p. 724)	Returns the Virtuozzo Tools package state information.
	PrlVmToolsInfo_GetVersion (p. 725)	Returns version information of the Virtuozzo Tools package installed in a given virtual machine.

PrlVmToolsInfo_GetState Function

Returns the Virtuozzo Tools package state information.

Syntax

```
PRL_RESULT PrlVmToolsInfo_GetState(
    PRL_HANDLE handle,
    PRL_VM_TOOLS_STATE_PTR pVmToolsState
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_TOOLS_INFO (p. 724).

pVmToolsState

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_NO_DATA (p. 1324) - the Virtuozzo Tools package state information is not available.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmToolsInfo_GetVersion Function

Returns version information of the Virtuozzo Tools package installed in a given virtual machine.

Syntax

```
PRL_RESULT PrlVmToolsInfo_GetVersion(
    PRL_HANDLE handle,
    PRL_STR sName,
    PRL_UINT32_PTR pnNameBufLength
);
```

File

PrlApiVm.h

Parameters

handle

A handle of type PHT_VM_TOOLS_INFO (p. 724).

sName

[out] A pointer to a buffer that receives the version information (a UTF-8 encoded, null-terminated string).

pnNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_NO_DATA (p. 1324) - the Virtuozzo Tools package version information is not available.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Before calling this function, make sure that the Virtuozzo Tools package is installed in the virtual machine by calling the PrlVmToolsInfo_GetState (p. 724) function.

PHT_VM_VIRTUAL_DEVICES_INFO

The PHT_VM_VIRTUAL_DEVICES_INFO handle is a container that contains information about the state of the virtual devices in a virtual machine. To obtain the handle, use the PrlVm_GetVirtDevInfo function.

Common Functions

This chapter contains information about common API functions.

Topics

Name	Description
System Functions (p. 727)	Common Virtuozzo API system functions.
API Debug Functions (p. 744)	API debug functions.
Handle Functions (p. 747)	Functions that perform operations on handles.
Virtual Device Functions (p. 752)	This section describes common virtual device management functions.
PrlBuffer (p. 775)	

System Functions

Common Virtuozzo API system functions.

Functions

	Name	Description
	PrlApi_CreateHandlesList (p. 728)	Creates a new handle of type PRL_HANDLES_LIST.
	PrlApi_CreateOpTypeList (p. 729)	Creates a new handle of type PHT_OPAQUE_TYPE_LIST (p. 177).
	PrlApi_CreateProblemReport (p. 729)	Creates and returns a handle of type PHT_PROBLEM_REPORT (p. 192), which is used to obtain and manage Virtuozzo problem reports.
	PrlApi_CreateStringsList (p. 730)	Creates a new handle of type PHT_STRINGS_LIST (p. 357).
	PrlApi_Deinit (p. 731)	De-initializes the library.
	PrlApi_GetAppMode (p. 731)	Returns the Virtuozzo API application mode.
	PrlApi_GetCrashDumpsPath (p. 732)	Returns the name and path of the directory where Virtuozzo crash dumps are stored.
	PrlApi_GetDefaultOsVersion (p. 732)	Returns default guest OS version for specified guest OS type at current API mode (see PrlApi_InitEx (p. 739) for more info about API mode).
	PrlApi_GetMessageType (p. 733)	Evaluates the specified error code and returns its classification (warning, question, information, etc.).
	PrlApi_GetRecommendMinVmMem (p. 733)	The PrlApi_GetRecommendMinVmMem returns recommend minimal memory size for guest OS defined in the OS version parameter.
	PrlApi_GetResultDescription (p. 734)	Evaluates a return code and returns a description of the problem.
	PrlApi_GetSupportedOsesTypes (p. 735)	Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes types list for current API mode (see PrlApi_InitEx (p. 739) for more info about API mode).
	PrlApi_GetSupportedOsesVersions (p. 735)	Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes list for specified guest OS type at current API mode (see PrlApi_InitEx (p. 739) for more info about API mode).
	PrlApi_GetVersion (p. 736)	Returns the Virtuozzo API version number.
	PrlApi_GuestToString (p. 736)	Returns a readable string representation given guest OS ID.
	PrlApi_Init (p. 737)	Initializes Virtuozzo API library.

	PrlApi_InitCrashHandler (p. 739)	Initiates the standard Virtuozzo crash dump handler.
	PrlApi_InitEx (p. 739)	This is extended version of PrlApi_Init (p. 737) call that can be used to specify explicitly application mode in which the API library should function.
	PrlApi_MsgCanBeIgnored (p. 740)	The PrlApi_MsgCanBeIgnored function allows to evaluate an error code and determine if the error is critical (i.e.
	PrlApi_SendPackedProblemReport (p. 741)	The PrlApi_SendPackedProblemReport lets to send previously extracted with PrlSrv_GetPackedProblemReport (p. 252) or PrlVm_GetPackedProblemReport (p. 465) calls problem report to the Virtuozzo support server.
	PrlApi_SendProblemReport (p. 742)	The PrlApi_SendProblemReport lets to send previously extracted with PrlSrv_GetProblemReport (p. 254) or PrlVm_GetProblemReport (p. 466) calls problem report to the Virtuozzo support server.
	PrlApi_SwitchConsoleLogging (p. 743)	Turn on/off print logging messages on console • PRL_ERR_SUCCESS (p. 1365) - function completed successfully
	PrlAppliance_Create (p. 743)	Creates a new handle of type PHT_APPLIANCE_CONFIG.

Topics

Name	Description
PrlApi_AddRemoteDevCommandBuffer@PRL_HANDLE@PRL_VOID_PTR@PRL_UINT32 (p. 728)	
PrlApi_AddRemoteDevCommandParam@PRL_HANDLE@PRL_UINT32 (p. 728)	

PrlApi_AddRemoteDevCommandBuffer@PRL_HANDLE@PRL_VOID_PTR@PRL_UINT32

PrlApi_AddRemoteDevCommandParam@PRL_HANDLE@PRL_UINT32

PrlApi_CreateHandlesList Function

Creates a new handle of type PRL_HANDLES_LIST.

Syntax

```
PRL_RESULT PrlApi_CreateHandlesList(
    PRL_HANDLE_PTR phHandlesList
);
```

File

PrlApiCore.h

Parameters

phHandlesList

A pointer to a variable that receives the new handle.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - inavlid (NULL) pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the new handle.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

PrlApi_CreateOpTypeList Function

Creates a new handle of type PHT_OPAQUE_TYPE_LIST (p. 177).

Syntax

```
PRL_RESULT PrlApi_CreateOpTypeList(
    PRL_HANDLE_PTR phOpTypeList,
    PRL_SIZE nTypeSize
);
```

File

PrlApiCore.h

Parameters

phOpTypeList

[out] A pointer to a variable that receives the handle.

nTypeSize

size of opaque type for which list should be instantiated

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed or size of not supported opaque type was passed. In order to determine list of supported opaque types please refer with PrlTypes.h.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Simple sample of method usage follow:

PrlApi_CreateOpTypeList(&hOpTypeList, sizeof(PRL_UINT32 (p. 798)));

PrlApi_CreateProblemReport Function

Creates and returns a handle of type PHT_PROBLEM_REPORT (p. 192), which is used to obtain and manage Virtuozzo problem reports.

Syntax

```
PRL_RESULT PrlApi_CreateProblemReport(
    PRL_PROBLEM_REPORT_SCHEME nReportScheme,
    PRL_HANDLE_PTR phProblemReport
);
```

File

PrlApiCore.h

Parameters

nReportScheme

[in] the type of report that will be generated (XML based or new packed form).

phProblemReport

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or unknown report scheme was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the problem report object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_CreateStringsList Function

Creates a new handle of type PHT_STRINGS_LIST (p. 357).

Syntax

```
PRL_RESULT PrlApi_CreateStringsList(
    PRL_HANDLE_PTR phStringsList
);
```

File

PrlApiCore.h

Parameters

phStringsList

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the handle.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_Deinit Function

De-initializes the library.

Syntax

```
PRL_RESULT PrlApi_Deinit();
```

File

PrlApiCore.h

Returns

PRL_RESULT. Possible values:

PRL_ERR_UNINITIALIZED (p. 1373) - the library was not initialized.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

Before exiting a client program, this function must be called to perform internal library de-initialization. Internally, this call will lead to the destruction of the internal event-loop and de-initialization of the library state.

See Also

PrlApi_Init (p. 737)

PrlApi_GetAppMode Function

Returns the Virtuozzo API application mode.

Syntax

```
PRL_RESULT PrlApi_GetAppMode(
    PRL_APPLICATION_MODE_PTR app_mode
);
```

File

PrlApiCore.h

Parameters

app_mode

[out] Current application mode of the API.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetCrashDumpsPath Function

Returns the name and path of the directory where Virtuozzo crash dumps are stored.

Syntax

```
PRL_RESULT PrlApi_GetCrashDumpsPath(
    PRL_STR sPath,
    PRL_UINT32_PTR pnPathBufLength
);
```

File

PrlApiCore.h

Parameters

sPath

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnPathBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer to the buffer size was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetDefaultOsVersion Function

Returns default guest OS version for specified guest OS type at current API mode (see PrlApi_InitEx (p. 739) for more info about API mode).

Syntax

```
PRL_RESULT PrlApi_GetDefaultOsVersion(
    PRL_UINT8 nGuestOsType,
    PRL_UINT16_PTR pnOsVersion
);
```

File

PrlApiVm.h

Parameters

nGuestOsType

requesting guest OS type. In order to determine list of supported guests types please refer PrlOses.h

pnOsVersion

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or unknown OS type was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetMessageType Function

Evaluates the specified error code and returns its classification (warning, question, information, etc.).

Syntax

```
PRL_RESULT PrlApi_GetMessageType(
    PRL_RESULT nErrCode,
    PRL_MESSAGE_TYPE_ENUM_PTR pnMsgType
);
```

File

PrlApiCore.h

Parameters

nErrCode

The error code to evaluate.

pnMsgType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetRecommendMinVmMem Function

The PrlApi_GetRecommendMinVmMem returns recommend minimal memory size for guest OS defined in the OS version parameter.

Syntax

```
PRL_RESULT PrlApi_GetRecommendMinVmMem(
    PRL_UINT32 nOsVersion,
    PRL_UINT32_PTR pnMinMemSize
);
```

File

PrlApiVm.h

Parameters

nOsVersion

OS version

pnMinMemSize

[out] A pointer to a variable that receives the result. Result is the recommend minimal memory size for guest OS.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetResultDescription Function

Evaluates a return code and returns a description of the problem.

Syntax

```
PRL_RESULT PrlApi_GetResultDescription(
    PRL_RESULT nErrCode,
    PRL_BOOL bIsBriefMessage,
    PRL_BOOL bFormated,
    PRL_STR sErrString,
    PRL_UINT32_PTR pnErrStringBufLength
);
```

File

PrlApiCore.h

Parameters

nErrCode

A return code to get the description for.

bIsBriefMessage

Specifies whether to obtain a long or a brief description of the problem. PRL_TRUE (p. 1652) specifies to obtain a brief message. PRL_FALSE (p. 1626) specifies to obtain a long message.

bFormated

Specifies whether HTML tags used for text formatting should be included in the resulting string.

sErrString

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnErrStringBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

PRL_TRUE

include HTML tags; PRL_FALSE (p. 1626) - do not include the tags.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

All synchronous functions in the Virtuozzo API return PRL_RESULT - a numeric return code used to evaluate the success or failure of the operation. If the return code indicates a failure, the PrlApi_GetResultDescription function can be used to obtain the description of the problem.

PrlApi_GetSupportedOsesTypes Function

Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes types list for current API mode (see PrlApi_InitEx (p. 739) for more info about API mode).

Syntax

```
PRL_RESULT PrlApi_GetSupportedOsesTypes(
    PRL_HOST_OS_TYPE nHostOsType,
    PRL_HANDLE_PTR phOsesTypesList
);
```

File

PrlApiVm.h

Parameters

nHostOsType

type of host system (remote or local) for which supported OSes list should be retrieved. In remote work mode please determine at first remote side host type (please use PrlSrvCfg_GetHostOsType (p. 330) for this). In local work mode PHO_UNKNOWN can be passed - host OS type will be automatically detected by API call in this case.

phOsesTypesList

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetSupportedOsesVersions Function

Returns handle of type PHT_OPAQUE_TYPE_LIST (p. 177) which contains supported guest OSes list for specified guest OS type at current API mode (see PrlApi_InitEx (p. 739) for more info about API mode).

Syntax

```
PRL_RESULT PrlApi_GetSupportedOsesVersions(
    PRL_HOST_OS_TYPE nHostOsType,
    PRL_UINT8 nGuestOsType,
    PRL_HANDLE_PTR phGuestsList
);
```

File

PrlApiVm.h

Parameters

nHostOsType

type of host system (remote or local) for which supported OSes list should be retrieved. In remote work mode please determine at first remote side host type (please use PrlSrvCfg_GetHostOsType (p. 330) for this). In local work mode PHO_UNKNOWN can be passed - host OS type will be automatically detected by API call in this case.

nGuestOsType

requesting guest OS type. In order to determine list of supported guests types please refer PrlOses.h

phOsesTypesList

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer or unknown OS type was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create the list object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_GetVersion Function

Returns the Virtuozzo API version number.

Syntax

```
PRL_RESULT PrlApi_GetVersion(
    PRL_UINT32_PTR current_version
);
```

File

PrlApiCore.h

Parameters

current_version

[out] Current version of the API.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Version number is incremented each time a public interfaces is changed. Our goal is to achieve full backward compatibility where the changes add new functionality without breaking the existing one.

PrlApi_GuestToString Function

Returns a readable string representation given guest OS ID.

Syntax

```
PRL_RESULT PrlApi_GuestToString(
    PRL_UINT32 nGuestId,
    PRL_STR sGuestString,
    PRL_UINT32_PTR pnGuestStringSize
);
```

File

PrlApiCore.h

Parameters

nGuestId

[in] Guest OS ID

sGuestString

[out] Guest OS name as string.

pnPathStringSize

[in] The size of string containing guest OS name

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer to the string size was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_Init Function

Initializes Virtuozzo API library.

Syntax

```
PRL_RESULT PrlApi_Init(
    PRL_UINT32 version
);
```

File

PrlApiCore.h

Returns

PRL_RESULT. Possible values:

PRL_ERR_DOUBLE_INIT (p. 1267) - library has been already initialized.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to initialize the library.

PRL_ERR_API_INCOMPATIBLE (p. 1183) - SDK version is incompatible with version in param.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

This function must be called after the dynamic libraries are loaded and before making any of the API calls. Internally, this call creates the internal event-loop and initializes the library state. Before exiting the program, the PrlApi_Deinit (p. 731) function must be called to perform internal library de-initialization.

Version

Library version number. This should be PARALLELS_API_VER (p. 1669).

See Also

PrlApi_Deinit (p. 731)

Example

The following example illustrates how to perform the necessary API library initialization.

The first step is to load the dynamic link library using the SdkWrap_Load function from the supplied wrapper library (SdkWrap is a helper library that contains just two functions for loading and unloading the API dynamic link library; the library is not documented in this guide).

The second step is to initialize the Virtuozzo API using the PrlApi_Init function.

Once the above steps are completed, you are ready to make other API calls, such as PrlSrv_Login (p. 272) and others.

On program exit, you must cal the PrlApi_Deinit (p. 731) function to deinitialize the API, and the SdkWrap_Unload function to unload the dynamic link libraries.

```
int main(int argc, char* argv[])
{
   // Pick the correct dynamic library depending on the platform.
   #ifdef _WIN_
      #define SDK_LIB_NAME "prl_sdk.dll"
   #elif defined(_LIN_)
      #define SDK_LIB_NAME "libprl_sdk.so"
   #elif defined(_MAC_)
      #define SDK_LIB_NAME "libprl_sdk.dylib"
   #endif

   // Load the dynamic library.
   // If the library cannot be found, try searching for it.
   if (PRL_FAILED(SdkWrap_Load(SDK_LIB_NAME)) &&
      PRL_FAILED(SdkWrap_Load("./" SDK_LIB_NAME)))
   {
      fprintf(stderr, "Failed to load " SDK_LIB_NAME "\n");
      return -1;
   }

   // Initialize the API.
   PRL_RESULT ret;
   ret = PrlApi_Init(PARALLELS_API_VER);
   if (PRL_FAILED(ret))
   {
      printf("PrlApi_Init returned with error (%s)\n",
      PRL_RESULT_TO_STRING(ret));
      return -1;
   }

   // your program code goes here...

   // Deinitialize the API.
   ret = PrlApi_Deinit();
   if (PRL_FAILED(ret))
   {
      printf("PrlApi_Deinit returned with error (%s)\n",
      PRL_RESULT_TO_STRING(ret));
      return -1;
   }

   // Unload dynamic link libraries.
   SdkWrap_Unload();
```

```
    return 0;
}
```

PrlApi_InitCrashHandler Function

Initiates the standard Virtuozzo crash dump handler.

Syntax

```
PRL_RESULT PrlApi_InitCrashHandler(
    PRL_CONST_STR sCrashDumpFileSuffix
);
```

File

PrlApiCore.h

Parameters

sCrashDumpFileSuffix

A user-define suffix to add to the crash dump file name (UTF-8 encoded, null-terminated string). The parameter is optional and can be NULL.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Call this function if you would like a dump file to be created on the application crash. The crush dump will be created in the common Virtuozzo crash dump directory (use PrlApi_GetCrashDumpsPath (p. 732) to obtains its name and path). The crash dump file name will begin with the name of the crashed application and can also contain a user-define suffix that is passed a parameter (see the Parameters subsection).

PrlApi_InitEx Function

This is extended version of PrlApi_Init (p. 737) call that can be used to specify explicitly application mode in which the API library should function.

Syntax

```
PRL_RESULT PrlApi_InitEx(
    PRL_UINT32 nVersion,
    PRL_APPLICATION_MODE nAppMode,
    PRL_UINT32 nFlags,
    PRL_UINT32 nReserved
);
```

File

PrlApiCore.h

Parameters

nVersion

Library version number. This should be PARALLELS_API_VER (p. 1669).

nAppMode

API library application mode.

nFlags

Flags as mask (from enums PRL_API_INIT_FLAGS (p. 1038) and PRL_API_COMMAND_FLAGS (p. 1037))

nReserved

reserved parameter for future purposes.

Returns

PRL_RESULT. Possible values:

PRL_ERR_DOUBLE_INIT (p. 1267) - library has been already initialized.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to initialize the library.

PRL_ERR_API_INCOMPATIBLE (p. 1183) - SDK version is incompatible with version in param.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

See Also

PrlApi_Deinit (p. 731) PrlApi_Init (p. 737)

PrlApi_MsgCanBeIgnored Function

The PrlApi_MsgCanBeIgnored function allows to evaluate an error code and determine if the error is critical (i.e.

Syntax

```
PRL_RESULT PrlApi_MsgCanBeIgnored(
    PRL_RESULT nErrCode,
    PRL_BOOL_PTR pbCanBeIgnored
);
```

File

PrlApiCore.h

Parameters

nErrCode

Error code to evaluate.

pbCanBeIgnored

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the error is not critical. PBL_FALSE indicates that the error is critical and should be properly handled.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

whether it can be ignored).

PrlApi_SendPackedProblemReport Function

The PrlApi_SendPackedProblemReport lets to send previously extracted with PrlSrv_GetPackedProblemReport (p. 252) or PrlVm_GetPackedProblemReport (p. 465) calls problem report to the Virtuozzo support server.

Syntax

```
PRL_HANDLE PrlApi_SendPackedProblemReport(
    PRL_HANDLE hProblemReport,
    PRL_BOOL bUseProxy,
    PRL_CONST_STR sProxyHost,
    PRL_UINT32 nProxyPort,
    PRL_CONST_STR sProxyUserLogin,
    PRL_CONST_STR sProxyUserPasswd,
    PRL_UINT32 nProblemSendTimeout,
    PRL_UINT32 nReserved,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR pUserData
);
```

File

PrlApiCore.h

Parameters

hProblemReport

handle of PHT_EVENT (p. 72) type contains packed problem report data

bUseProxy

sign specifies whether proxy settings should be used. Either system proxy settings can be automatically by API call retrieved and used or custom settings can be specified by follow params (note that custom values override correspond system settings)

sProxyHost

[optional] if using proxy this param lets to specify proxy server host address/name (can be null or empty)

nProxyPort

[optional] if using proxy this param lets to specify proxy server port (should be 0 in order to specify to use system settings)

sProxyUserLogin

[optional] if using proxy this param lets to specify user login if last one not specified at system proxy settings or need to be overridden (can be null or empty)

sProxyUserPasswd

[optional] if using proxy this param lets to specify user password if last one not specified at system proxy settings (can be null or empty)

nProblemSendTimeout

timeout in milliseconds which specifies maximal awaiting problem report send time (0 value not limiting problem report send operation at all)

nReserved

[unused] reserved parameter for further purposes

handler

[optional] A pointer to a user callback function (event handler) that will be called every time a problem report sending progress changed or job completed.

pUserData

[optional] A pointer to a user data that will be passed to the optional callback function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_SendProblemReport Function

The PrlApi_SendProblemReport lets to send previously extracted with PrlSrv_GetProblemReport (p. 254) or PrlVm_GetProblemReport (p. 466) calls problem report to the Virtuozzo support server.

Syntax

```
PRL_HANDLE PrlApi_SendProblemReport(
    PRL_CONST_STR sProblemReport,
    PRL_BOOL bUseProxy,
    PRL_CONST_STR sProxyHost,
    PRL_UINT32 nProxyPort,
    PRL_CONST_STR sProxyUserLogin,
    PRL_CONST_STR sProxyUserPasswd,
    PRL_UINT32 nProblemSendTimeout,
    PRL_UINT32 nReserved,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR pUserData
);
```

File

PrlApiCore.h

Parameters

sProblemReport

problem report string representation (expected null terminated string in UTF-8 encoding)

bUseProxy

sign specifies whether proxy settings should be used. Either system proxy settings can be automatically by API call retrieved and used or custom settings can be specified by follow params (note that custom values override correspond system settings)

sProxyHost

[optional] if using proxy this param lets to specify proxy server host address/name (can be null or empty)

nProxyPort

[optional] if using proxy this param lets to specify proxy server port (should be 0 in order to specify to use system settings)

sProxyUserLogin

[optional] if using proxy this param lets to specify user login if last one not specified at system proxy settings or need to be overridden (can be null or empty)

sProxyUserPasswd

[optional] if using proxy this param lets to specify user password if last one not specified at system proxy settings (can be null or empty)

nProblemSendTimeout

timeout in milliseconds which specifies maximal awaiting problem report send time (0 value not limiting problem report send operation at all)

nReserved

[unused] reserved parameter for further purposes

handler

[optional] A pointer to a user callback function (event handler) that will be called every time a problem report sending progress changed or job completed.

pUserData

[optional] A pointer to a user data that will be passed to the optional callback function.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlApi_SwitchConsoleLogging Function

Turn on/off print logging messages on console

- PRL_ERR_SUCCESS (p. 1365) - function completed successfully

Syntax

```
PRL_RESULT PrlApi_SwitchConsoleLogging(
    PRL_BOOL bSwitchOn
);
```

File

PrlApiCore.h

PrlAppliance_Create Function

Creates a new handle of type PHT_APPLIANCE_CONFIG.

Syntax

```
PRL_RESULT PrlAppliance_Create(
    PRL_HANDLE_PTR phAppCfg
);
```

File

PrlApiVm.h

Parameters

phAppCfg

[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - null pointer was passed.

PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory to create a virtual network object.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The handle is necessary to create and configure an appliance download configuration. If you would like to create a new appliance download configuration, use this function to create the handle first and then use the handle functions to configure the appliance info.

It is the responsibility of the caller to free the handle with the PrlHandle_Free (p. 748) call when it is no longer needed.

API Debug Functions

API debug functions.

Functions

	Name	Description
	event_type_to_string (p. 744)	This inline function returns a readable string representation of the specified event type.
	handle_type_to_string (p. 745)	This inline function returns a readable string representation of the specified handle type.
	prl_result_to_string (p. 745)	This inline function returns a readable string representation of the PRL_RESULT value.
	PrlDbg_EventTypeToString (p. 745)	Returns a readable string representation of the specified event type.
	PrlDbg_GetHandlesNum (p. 746)	The PrlDbg_GetHandlesNum function determines how many handles were instantiated in the API library.
	PrlDbg_HandleTypeToString (p. 746)	Returns a readable string representation of the specified handle type.
	PrlDbg_PrlResultToString (p. 747)	Returns a readable string representation of the PRL_RESULT value.

event_type_to_string Function

This inline function returns a readable string representation of the specified event type.

Syntax

```
static inline const char * event_type_to_string(
    PRL_EVENT_TYPE value
);
```

File

PrlApiCore.h

Parameters

value

The event type.

handle_type_to_string Function

This inline function returns a readable string representation of the specified handle type.

Syntax

```
static inline const char * handle_type_to_string(
    PRL_HANDLE_TYPE value
);
```

File

PrlApiCore.h

Parameters

value

The handle type.

prl_result_to_string Function

This inline function returns a readable string representation of the PRL_RESULT value.

Syntax

```
static inline const char * prl_result_to_string(
    PRL_RESULT value
);
```

File

PrlApiCore.h

Parameters

value

A variable of type PRL_RESULT.

PrlDbg_EventTypeToString Function

Returns a readable string representation of the specified event type.

Syntax

```
PRL_RESULT PrlDbg_EventTypeToString(
    PRL_EVENT_TYPE event_type,
    PRL_CONST_STR_PTR result
);
```

File

PrlApiCore.h

Parameters

event_type

Event type.

result

[out] Event type as a string.

Returns

PRL_ERR_SUCCESS (p. 1365)

Remarks

Helpful for logging.

PrlDbg_GetHandlesNum Function

The PrlDbg_GetHandlesNum function determines how many handles were instantiated in the API library.

Syntax

```
PRL_RESULT PrlDbg_GetHandlesNum(
    PRL_UINT32_PTR handles_num,
    PRL_HANDLE_TYPE type
);
```

File

PrlApiCore.h

Parameters

handles_num

[out] A pointer to a variable that receives the result.

type

specifies type of handles which instances num necessary to return. If this param has value PHT_ERROR then handle type is ignored and method returns common handles num. Note: using this method in case to get num of handles instances of concrete type may suspends SDK lib functionality on a quite long time.

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid (NULL) pointer for storing result was passed PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlDbg_HandleTypeToString Function

Returns a readable string representation of the specified handle type.

Syntax

```
PRL_RESULT PrlDbg_HandleTypeToString(
    PRL_HANDLE_TYPE handle_type,
    PRL_CONST_STR_PTR result
);
```

File

PrlApiCore.h

Parameters

handle_type

Handle type.

result

[out] Handle type as a string.

Returns

PRL_ERR_SUCCESS (p. 1365)

Remarks

Helpful for logging.

PrlDbg_PrlResultToString Function

Returns a readable string representation of the PRL_RESULT value.

Syntax

```
PRL_RESULT PrlDbg_PrlResultToString(
    PRL_RESULT value,
    PRL_CONST_STR_PTR result
);
```

File

PrlApiCore.h

Parameters

value

A variable of type PRL_RESULT.

result

[out] The PRL_RESULT value as a string.

Returns

PRL_ERR_SUCCESS (p. 1365)

Remarks

Helpful for logging.

Handle Functions

Functions that perform operations on handles.

Functions

	Name	Description
	PrlHandle_AddRef (p. 748)	Increases the specified handle reference count by one.
	PrlHandle_Free (p. 748)	Frees the specified handle.

	PrlHandle_GetChannelStatistics (p. 749)	Retrieves IO channel statistics (throughput, delay) for correspond virtual machine or remote desktop control object.
	PrlHandle_GetPackageId (p. 750)	Returns the numeric identifier of the package that caused the event.
	PrlHandle_GetType (p. 750)	Returns the type of the specified handle.
	PrlHandle_RegEventHandler (p. 751)	Registers event handler.
	PrlHandle_UnregEventHandler (p. 751)	Unregisters event handler.

PrlHandle_AddRef Function

Increases the specified handle reference count by one.

Syntax

```
PRL_RESULT PrlHandle_AddRef(
    PRL_HANDLE handle
);
```

File

PrlApiCore.h

Parameters

handle

Handle to duplicate.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Handles used within the Virtuozzo API are thread safe. They can be used in multiple threads at the same time. To maintain the proper reference counting, the count should be increased each time a handle is passed to another thread by calling the PrlHandle_AddRef function. If this is not done, freeing a handle in one thread may destroy it while other threads are still using it.

See Also

PrlHandle_Free (p. 748)

PrlHandle_Free Function

Frees the specified handle.

Syntax

```
PRL_RESULT PrlHandle_Free(
    PRL_HANDLE handle
);
```

File

PrlApiCore.h

Parameters

handle

Handle to free.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Handles used within the Virtuozzo API are reference counted. Each handle contains a count of the number of references to it held by other objects. A corresponding object stays in memory for as long as the count is greater than zero. A client application is responsible for freeing any handles that are no longer required. A handle can be freed using the PrlHandle_Free function. The function decreases the reference count by one. When the count reaches zero, the object is destroyed. Failing to free a handle after it has been used will result in a memory leak.

PrlHandle_GetChannelStatistics Function

Retrieves IO channel statistics (throughput, delay) for correspond virtual machine or remote desktop control object.

Syntax

```
PRL_RESULT PrlHandle_GetChannelStatistics(
    PRL_HANDLE hObj,
    PRL_IO_CHANNEL_STAT_PTR channelStat
);
```

File

PrlApiCore.h

Parameters

hObj

A handle of type PHT_SERVER (p. 219), PHT_VIRTUAL_MACHINE (p. 442) or PHT_DESKTOP_CONTROL (p. 32).

channelStat

A pointer to the statistics structure

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed. PRL_ERR_NO_DATA (p. 1324) - mostly indicates there is no connection to remote desktop now available. I.e. PrlDevDisplay_ConnectToVm was not called.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlHandle_GetPackageId Function

Returns the numeric identifier of the package that caused the event.

Syntax

```
PRL_RESULT PrlHandle_GetPackageId(
    PRL_HANDLE hObj,
    PRL_UINT64_PTR id
);
```

File

PrlApiCore.h

Parameters

id

[out] A pointer to a variable that receives the result (event id).

hEvent

A handle of type PHT_EVENT (p. 72) or PHT_JOB (p. 138).

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

The numeric identifier is a monotonically increasing number. It is reset to one on dispatcher restart. If event or job was not originated in the dispatcher, then the numeric identifier is equal to zero. A triplet of values forms a unique monotonically increasing numeric identifier that could be used for event ordering. See PrlSrvInfo_GetStartTime (p. 348) and PrlSrvInfo_GetStartTimeMonotonic (p. 349) for details on how to obtain (monotonic) dispatcher start time value.

See Also

PrlSrvInfo_GetStartTime (p. 348) PrlSrvInfo_GetStartTimeMonotonic (p. 349)

PrlHandle_GetType Function

Returns the type of the specified handle.

Syntax

```
PRL_RESULT PrlHandle_GetType(
    PRL_HANDLE handle,
    PRL_HANDLE_TYPE_PTR type
);
```

File

PrlApiCore.h

Parameters

handle

Handle to get the type for.

type

[out] A pointer to a variable that receives the result. The type is returned as an enumerator of the PRL_HANDLE_TYPE (p. 1064) enumeration.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Handle is an abstraction that is used to hide an object implementation. While there are many methods used to access functionality of different objects, this method is used to identify object type as one of the PRL_HANDLE_TYPE (p. 1064) types.

PrlHandle_RegEventHandler Function

Registers event handler.

Syntax

```
PRL_RESULT PrlHandle_RegEventHandler(
    PRL_HANDLE hObj,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR userData
);
```

File

PrlApiCore.h

Parameters

hObj

[in] A handle of type PHT_SERVER (p. 219), PHT_VIRTUAL_MACHINE (p. 442), PHT_DESKTOP_CONTROL (p. 32) or PHT_CVSRC (p. 32).

handler

[in] User defined function that will be called for each event.

userData

[in] User defined data that will be passed to handler.

Returns

PRL_RESULT. Possible values: PRL_ERR_UNINITIALIZED (p. 1373) - SDK was not properly initialized. PRL_ERR_INVALID_ARG (p. 1290) - specified arguments are invalid. PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlHandle_UnregEventHandler Function

Unregisters event handler.

Syntax

```
PRL_RESULT PrlHandle_UnregEventHandler(
    PRL_HANDLE hObj,
    PRL_EVENT_HANDLER_PTR handler,
    PRL_VOID_PTR userData
);
```

File

PrlApiCore.h

Parameters

hObj

[in] A handle of type PHT_SERVER (p. 219), PHT_VIRTUAL_MACHINE (p. 442), PHT_DESKTOP_CONTROL (p. 32) or PHT_CVSRC (p. 32).

handler

[in] User defined function specified in previous call to PrlHandle_RegEventHandler (p. 751)().

userData

[in] User defined data specified in previous call to PrlHandle_RegEventHandler (p. 751)().

Returns

PRL_RESULT. Possible values: PRL_ERR_UNINITIALIZED (p. 1373) - SDK was not properly initialized. PRL_ERR_INVALID_ARG (p. 1290) - specified arguments are invalid. PRL_ERR_OUT_OF_MEMORY (p. 1334) - not enough memory. PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Virtual Device Functions

This section describes common virtual device management functions.

Description

Common virtual device management functions can operate on virtual devices of the following types:

`PHT_VIRTUAL_DEV_FLOPPY ( p. 381)`

`PHT_VIRTUAL_DEV_HARD_DISK ( p. 381)`

`PHT_VIRTUAL_DEV_NET_ADAPTER ( p. 399)`

`PHT_VIRTUAL_DEV_OPTICAL_DISK ( p. 425)`

`PHT_VIRTUAL_DEV_PARALLEL_PORT ( p. 425)`

`PHT_VIRTUAL_DEV_SERIAL_PORT ( p. 425)`

`PHT_VIRTUAL_DEV_SOUND ( p. 427)`

`PHT_VIRTUAL_DEV_USB_DEVICE ( p. 430)`

Each common function accepts a handle of type `PRL_HANDLE ( p. 820)` identifying the virtual device to operate on. The actual handle passed can be of any type from the list above. In addition to common function described here, most of the individual virtual devices have functions specific to their respective types. For device-type specific functions, follow links to virtual device handles in the list above.

Functions

	Name	Description
	PrlVmDev_Connect (p. 754)	The PrlVmDev_Connect function allows to connect virtual devices while the virtual machine is running.
	PrlVmDev_CopyImage (p. 754)	Copies a virtual device image to another custom location.
	PrlVmDev_Create (p. 755)	Creates a new virtual device handle not bound to any virtual machine.
	PrlVmDev_CreateImage (p. 755)	Physically creates a virtual device image on the host.
	PrlVmDev_Disconnect (p. 756)	Allows to disconnect a device while the virtual machine is running.
	PrlVmDev_GetDescription (p. 757)	Returns the description of a specified virtual device.
	PrlVmDev_GetEmulatedType (p. 757)	Returns the virtual device emulation type.
	PrlVmDev_GetFriendlyName (p. 758)	Returns the virtual device user friendly name.
	PrlVmDev_GetIfaceType (p. 759)	Returns the virtual device interface type (IDE or SCSI).
	PrlVmDev_GetImagePath (p. 760)	Returns virtual device image path.
	PrlVmDev_GetIndex (p. 760)	Returns an index identifying the virtual device.
	PrlVmDev_GetOutputFile (p. 761)	Returns the virtual device output file.
	PrlVmDev_GetStackIndex (p. 761)	Returns the virtual device stack index (position at the controller (IDE or SCSI) bus).
	PrlVmDev_GetSubType (p. 762)	Returns the virtual device subtype.
	PrlVmDev_GetSysName (p. 763)	Returns the virtual device system name.
	PrlVmDev_GetType (p. 763)	Returns the type of the specified device.
	PrlVmDev_IsConnected (p. 764)	Checks if the specified virtual device is connected.
	PrlVmDev_IsEnabled (p. 764)	Determines if the specified device is enabled.
	PrlVmDev_IsPassthrough (p. 765)	Determines if the passthrough mode is enabled for the specified mass storage device.
	PrlVmDev_IsRemote (p. 765)	Determines if the specified virtual device is a remote device.
	PrlVmDev_Remove (p. 766)	Removes the specified virtual device handle from the parent virtual machine list.
	PrlVmDev_ResizeImage (p. 766)	Resize a virtual device image.
	PrlVmDev_SetConnected (p. 767)	Connects the specified virtual device.
	PrlVmDev_SetDefaultStackIndex (p. 768)	Automatically generates and assigns a stack index to a virtual device (a corresponding device interface, IDE or SCSI, has to be set in advance).
	PrlVmDev_SetDescription (p. 768)	Sets the description for a specified virtual device.
	PrlVmDev_SetEmulatedType (p. 769)	Sets the specified virtual device emulation type.
	PrlVmDev_SetEnabled (p. 770)	Enables the specified virtual device..
	PrlVmDev_SetFriendlyName (p. 770)	Sets virtual device user-friendly name.
	PrlVmDev_SetIfaceType (p. 771)	Sets the virtual device interface type (IDE or SCSI).
	PrlVmDev_SetImagePath (p. 771)	Sets the virtual device image path.
	PrlVmDev_SetIndex (p. 772)	Sets an index identifying the virtual device.
	PrlVmDev_SetOutputFile (p. 772)	Sets the virtual device output file.
	PrlVmDev_SetPassthrough (p. 773)	Enables the passthrough mode for the specified mass storage device (optical ot hard disk).
	PrlVmDev_SetRemote (p. 773)	Changes the "remote" flag for the specified device.
	PrlVmDev_SetStackIndex (p. 774)	Sets the virtual device stack index (position at the IDE or SCSI controller bus).

	PrlVmDev_SetSubType (p. 774)	Sets the virtual device subtype.
	PrlVmDev_SetSysName (p. 775)	Sets virtual device system name.

PrlVmDev_Connect Function

The PrlVmDev_Connect function allows to connect virtual devices while the virtual machine is running.

Syntax

```
PRL_HANDLE PrlVmDev_Connect(
    PRL_HANDLE hVmDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_CopyImage Function

Copies a virtual device image to another custom location.

Syntax

```
PRL_HANDLE PrlVmDev_CopyImage(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewImageName,
    PRL_CONST_STR sTargetPath,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sNewImageName

New image short name.

sTargetPath
New custom location.

nFlags
Reserved

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_Create Function

Creates a new virtual device handle not bound to any virtual machine.

Syntax

```
PRL_RESULT PrlVmDev_Create(
    PRL_DEVICE_TYPE nDeviceType,
    PRL_HANDLE_PTR phVmDevice
);
```

File

PrlApiVm.h

Parameters

nDeviceType
The type of handle to create.

phVmDevice
[out] A pointer to a variable that receives the handle.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

It is necessary to call PrlHandle_Free (p. 748) after the handle usage.

PrlVmDev_CreateImage Function

Physically creates a virtual device image on the host.

Syntax

```
PRL_HANDLE PrlVmDev_CreateImage(
    PRL_HANDLE hVmDev,
    PRL_BOOL bRecreateIsAllowed,
    PRL_BOOL bNonInteractiveMode
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

bRecreateIsAllowed

Specifies wether the existing image should be replaced. If set to PRL_TRUE (p. 1652), the existing image will be replaced. If set to PRL_FALSE (p. 1626), the operation will be aborted if image already exists.

bNonInteractiveMode

Specifies wether the operation should be carried out in interactive mode (where the Virtuozzo Service may ask questions and the client will have to send answers). If set to PRL_TRUE (p. 1652), non-interactive mode will be used (the Virtuozzo Service will make decisions on its own). If set to PRL_FALSE (p. 1626), interactive mode will be used.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_Disconnect Function

Allows to disconnect a device while the virtual machine is running.

Syntax

```
PRL_HANDLE PrlVmDev_Disconnect(
    PRL_HANDLE hVmDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetDescription Function

Returns the description of a specified virtual device.

Syntax

```
PRL_RESULT PrlVmDev_GetDescription(
    PRL_HANDLE hVmDev,
    PRL_STR sDescription,
    PRL_UINT32_PTR pnDescriptionBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sDescription

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnDescriptionBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The pnDescriptionBufLength parameter will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDev_SetDescription (p. 768)

PrlVmDev_GetEmulatedType Function

Returns the virtual device emulation type.

Syntax

```
PRL_RESULT PrlVmDev_GetEmulatedType(
```

```
    PRL_HANDLE hVmDev,
    PRL_VM_DEV_EMULATION_TYPE_PTR pnEmulatedType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

pnEmulatedType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Emulation types are defined in the PRL_VM_DEV_EMULATION_TYPE (p. 944) enumeration. The following describes emulation types used for different types of virtual devices:

- PHT_VIRTUAL_DEV_FLOPPY (p. 381): PDT_USE_REAL_DEVICE, PDT_USE_IMAGE_FILE.
- PHT_VIRTUAL_DEV_HARD_DISK (p. 381): PDT_USE_REAL_DEVICE, PDT_USE_IMAGE_FILE.
- PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425): PDT_USE_REAL_DEVICE, PDT_USE_IMAGE_FILE.
- PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425): PDT_USE_REAL_DEVICE, PDT_USE_OUTPUT_FILE, PDT_USE_OTHER (for socket mode).
- PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425): PDT_USE_REAL_DEVICE, PDT_USE_OUTPUT_FILE, PDT_USE_OTHER (for printer mode).
- PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399): PDT_USE_REAL_DEVICE (host-only network), PDT_USE_IMAGE_FILE (shared network), PDT_USE_OUTPUT_FILE (bridget ethernet)
- PHT_VIRTUAL_DEV_SOUND (p. 427): PDT_USE_REAL_DEVICE (AC'97), PDT_USE_IMAGE_FILE (Creative).
- PHT_VIRTUAL_DEV_USB_DEVICE (p. 430): PDT_USE_REAL_DEVICE

PrlVmDev_GetFriendlyName Function

Returns the virtual device user friendly name.

Syntax

```
PRL_RESULT PrlVmDev_GetFriendlyName(
    PRL_HANDLE hVmDev,
    PRL_STR sFriendlyName,
    PRL_UINT32_PTR pnFriendlyNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sFriendlyName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnFriendlyNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetIfaceType Function

Returns the virtual device interface type (IDE or SCSI).

Syntax

```
PRL_RESULT PrlVmDev_GetIfaceType(
    PRL_HANDLE hVmDev,
    PRL_MASS_STORAGE_INTERFACE_TYPE_PTR pnIfaceType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425).

pnIfaceType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetImagePath Function

Returns virtual device image path.

Syntax

```
PRL_RESULT PrlVmDev_GetImagePath(
    PRL_HANDLE hVmDev,
    PRL_STR sImagePath,
    PRL_UINT32_PTR pnImagePathBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_FLOPPY (p. 381), PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425).

sImagePath

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnImagePathBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetIndex Function

Returns an index identifying the virtual device.

Syntax

```
PRL_RESULT PrlVmDev_GetIndex(
    PRL_HANDLE hVmDev,
    PRL_UINT32_PTR pnIndex
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_FLOPPY (p. 381),

PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425), PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425), PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399).

pnIndex

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetOutputFile Function

Returns the virtual device output file.

Syntax

```
PRL_RESULT PrlVmDev_GetOutputFile(
    PRL_HANDLE hVmDev,
    PRL_STR sOutputFile,
    PRL_UINT32_PTR pnOutputFileBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425), PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425).

sOutputFile

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnOutputFileBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetStackIndex Function

Returns the virtual device stack index (position at the controller (IDE or SCSI) bus).

Syntax

```
PRL_RESULT PrlVmDev_GetStackIndex(
    PRL_HANDLE hVmDev,
    PRL_UINT32_PTR pnStackIndex
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_GENERIC_SCSI, PHT_VIRTUAL_DEV_GENERIC_PCI.

pnStackIndex

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetSubType Function

Returns the virtual device subtype.

Syntax

```
PRL_RESULT PrlVmDev_GetSubType(
    PRL_HANDLE hVmDev,
    PRL_CLUSTERED_DEVICE_SUBTYPE_PTR pnSubType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425).

pnSubType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetSysName Function

Returns the virtual device system name.

Syntax

```
PRL_RESULT PrlVmDev_GetSysName(
    PRL_HANDLE hVmDev,
    PRL_STR sSysName,
    PRL_UINT32_PTR pnSysNameBufLength
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sSysName

[out] A pointer to a buffer that receives the result (a UTF-8 encoded, null-terminated string).

pnSysNameBufLength

[in] The size of the output buffer (in bytes). Set the buffer pointer to null and this parameter's value to zero to receive the required size. [out] The required output buffer size.

Returns

PRL_RESULT. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_BUFFER_OVERRUN (p. 1199) - the size of the output buffer is not large enough. The parameter that is used to specify the size will contain the required size.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_GetType Function

Returns the type of the specified device.

Syntax

```
PRL_RESULT PrlVmDev_GetType(
    PRL_HANDLE hVmDev,
    PRL_DEVICE_TYPE_PTR pnDeviceType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

pnDeviceType

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_IsConnected Function

Checks if the specified virtual device is connected.

Syntax

```
PRL_RESULT PrlVmDev_IsConnected(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbConnected
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

pbConnected

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - invalid device handle or non valid (NULL) pointer to storing result, or handle of already removed VM device was passed. PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmDev_IsEnabled Function

Determines if the specified device is enabled.

Syntax

```
PRL_RESULT PrlVmDev_IsEnabled(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbEnabled
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

pbEnabled

[out] A pointer to a variable that receives the result.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_IsPassthrough Function

Determines if the passthrough mode is enabled for the specified mass storage device.

Syntax

```
PRL_RESULT PrlVmDev_IsPassthrough(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbPassthrough
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the device. This can be one of the following: PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_HARD_DISK (p. 381).

pbPassthrough

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that the passthrough mode is enabled. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_IsRemote Function

Determines if the specified virtual device is a remote device.

Syntax

```
PRL_RESULT PrlVmDev_IsRemote(
    PRL_HANDLE hVmDev,
    PRL_BOOL_PTR pbRemote
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

pbRemote

[out] A pointer to a variable that receives the result. PRL_TRUE (p. 1652) indicates that this is a remote device. PRL_FALSE (p. 1626) indicates otherwise.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_Remove Function

Removes the specified virtual device handle from the parent virtual machine list.

Syntax

```
PRL_RESULT PrlVmDev_Remove(
    PRL_HANDLE hVmDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_ResizeImage Function

Resize a virtual device image.

Syntax

```
PRL_HANDLE PrlVmDev_ResizeImage(
    PRL_HANDLE hVmDev,
    PRL_UINT32 nNewSize,
    PRL_UINT32 nFlags
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

nNewSize

Specifies a new image size in Mbytes.

nFlags

bitset of (PRIF_DISK_INFO, PRIF_RESIZE_LAST_PARTITION). Also PACF_NON_INTERACTIVE_MODE can be used in order to specify non interactive session working mode (remote side won't ask questions during operation progress in this case).

Returns

A handle of type PHT_JOB (p. 138) containing the results of this asynchronous operation or PRL_INVALID_HANDLE (p. 1627) if there's not enough memory to instantiate the job object.

Remarks

To get the return code from the PHT_JOB (p. 138) object, use the PrlJob_GetRetCode (p. 143) function. Possible values are:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetConnected Function

Connects the specified virtual device.

Syntax

```
PRL_RESULT PrlVmDev_SetConnected(
    PRL_HANDLE hVmDev,
    PRL_BOOL bConnected
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

bConnected

Value to set. PRL_TRUE (p. 1652) to connect the device. PRL_FALSE (p. 1626) to disconnect it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetDefaultStackIndex Function

Automatically generates and assigns a stack index to a virtual device (a corresponding device interface, IDE or SCSI, has to be set in advance).

Syntax

```
PRL_RESULT PrlVmDev_SetDefaultStackIndex(
    PRL_HANDLE hVmDev
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_GENERIC_SCSI or PHT_VIRTUAL_DEV_GENERIC_PCI.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type was passed.

PRL_ERR_NO_MORE_FREE_INTERFACE_SLOTS (p. 1324) - there are no free slots left.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetDescription Function

Sets the description for a specified virtual device.

Syntax

```
PRL_RESULT PrlVmDev_SetDescription(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewDescription
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sNewDescription

The new description to set. The value must be a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

See Also

PrlVmDev_GetDescription (p. 757)

PrlVmDev_SetEmulatedType Function

Sets the specified virtual device emulation type.

Syntax

```
PRL_RESULT PrlVmDev_SetEmulatedType(
    PRL_HANDLE hVmDev,
    PRL_VM_DEV_EMULATION_TYPE nEmulatedType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

nEmulatedType

The emulation type to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

Remarks

Emulation types are defined in the PRL_VM_DEV_EMULATION_TYPE (p. 944) enumeration. The following describes emulation types used for different types of virtual devices:

- PHT_VIRTUAL_DEV_FLOPPY (p. 381): PDT_USE_REAL_DEVICE, PDT_USE_IMAGE_FILE.
- PHT_VIRTUAL_DEV_HARD_DISK (p. 381): PDT_USE_REAL_DEVICE, PDT_USE_IMAGE_FILE.
- PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425): PDT_USE_REAL_DEVICE, PDT_USE_IMAGE_FILE.
- PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425): PDT_USE_REAL_DEVICE, PDT_USE_OUTPUT_FILE, PDT_USE_OTHER (for socket mode).
- PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425): PDT_USE_REAL_DEVICE, PDT_USE_OUTPUT_FILE, PDT_USE_OTHER (for printer mode).
- PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399): PDT_USE_REAL_DEVICE (host only network), PDT_USE_IMAGE_FILE (shared network). PDT_USE_OUTPUT_FILE (bridget ethernet)
- PHT_VIRTUAL_DEV_SOUND (p. 427): PDT_USE_REAL_DEVICE (AC'97), PDT_USE_IMAGE_FILE (Creative).
- PHT_VIRTUAL_DEV_USB_DEVICE (p. 430): PDT_USE_REAL_DEVICE

PrlVmDev_SetEnabled Function

Enables the specified virtual device..

Syntax

```
PRL_RESULT PrlVmDev_SetEnabled(
    PRL_HANDLE hVmDev,
    PRL_BOOL bEnabled
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

bEnabled

Specify PRL_TRUE (p. 1652) to enable the device. Specified PRL_FALSE (p. 1626) to disable it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetFriendlyName Function

Sets virtual device user-friendly name.

Syntax

```
PRL_RESULT PrlVmDev_SetFriendlyName(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewFriendlyName
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sNewFriendlyName

The user-friendly name to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetIfaceType Function

Sets the virtual device interface type (IDE or SCSI).

Syntax

```
PRL_RESULT PrlVmDev_SetIfaceType(
    PRL_HANDLE hVmDev,
    PRL_MASS_STORAGE_INTERFACE_TYPE nIfaceType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_GENERIC_SCSI.

nIfaceType

The interface type to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or PMS_IDE_DEVICE for PHT_VIRTUAL_DEV_GENERIC_SCSI handle type was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetImagePath Function

Sets the virtual device image path.

Syntax

```
PRL_RESULT PrlVmDev_SetImagePath(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewImagePath
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_FLOPPY (p. 381), PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425).

sNewImagePath

The image path to set. The value must be a UTF-8 encoded, null-terminated string

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetIndex Function

Sets an index identifying the virtual device.

Syntax

```
PRL_RESULT PrlVmDev_SetIndex(
    PRL_HANDLE hVmDev,
    PRL_UINT32 nIndex
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_FLOPPY (p. 381), PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425), PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425), PHT_VIRTUAL_DEV_NET_ADAPTER (p. 399).

nIndex

The new index.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetOutputFile Function

Sets the virtual device output file.

Syntax

```
PRL_RESULT PrlVmDev_SetOutputFile(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewOutputFile
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_PARALLEL_PORT (p. 425), PHT_VIRTUAL_DEV_SERIAL_PORT (p. 425).

sNewOutputFile

Name and path of the new output file.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type, or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetPassthrough Function

Enables the passthrough mode for the specified mass storage device (optical ot hard disk).

Syntax

```
PRL_RESULT PrlVmDev_SetPassthrough(
    PRL_HANDLE hVmDev,
    PRL_BOOL bPassthrough
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the device. This can be one of the following: PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_HARD_DISK (p. 381).

bPassthrough

Specify PRL_TRUE (p. 1652) to enable the passthrough mode. Specify PRL_FALSE (p. 1626) to disab;e it.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetRemote Function

Changes the "remote" flag for the specified device.

Syntax

```
PRL_RESULT PrlVmDev_SetRemote(
    PRL_HANDLE hVmDev,
    PRL_BOOL bRemote
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

bRemote

Specify PRL_TRUE (p. 1652) to make the device appear as remote. Specify PRL_FALSE (p. 1626) to make the device appear as local.

Returns

PRL_RESULT. Possible values: PRL_ERR_INVALID_ARG (p. 1290) - non valid virtual device handle or handle of already removed VM device was passed PRL_ERR_SUCCESS (p. 1365) - operation completed successfully

PrlVmDev_SetStackIndex Function

Sets the virtual device stack index (position at the IDE or SCSI controller bus).

Syntax

```
PRL_RESULT PrlVmDev_SetStackIndex(
    PRL_HANDLE hVmDev,
    PRL_UINT32 nStackIndex
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_GENERIC_SCSI, PHT_VIRTUAL_DEV_GENERIC_PCI.

nStackIndex

A new index to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or invalid handle type was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetSubType Function

Sets the virtual device subtype.

Syntax

```
PRL_RESULT PrlVmDev_SetSubType(
    PRL_HANDLE hVmDev,
```

```
    PRL_CLUSTERED_DEVICE_SUBTYPE nSubType
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device. This can be one of the following: PHT_VIRTUAL_DEV_HARD_DISK (p. 381), PHT_VIRTUAL_DEV_OPTICAL_DISK (p. 425), PHT_VIRTUAL_DEV_GENERIC_SCSI.

nSubType

The subtype to set.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlVmDev_SetSysName Function

Sets virtual device system name.

Syntax

```
PRL_RESULT PrlVmDev_SetSysName(
    PRL_HANDLE hVmDev,
    PRL_CONST_STR sNewSysName
);
```

File

PrlApiVm.h

Parameters

hVmDev

A handle identifying the virtual device.

sNewSysName

The new system name to set. The value must be a UTF-8 encoded, null-terminated string.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid handle or null pointer was passed.

PRL_ERR_SUCCESS (p. 1365) - function completed successfully.

PrlBuffer

Functions

	Name	Description
	PrlBuffer_Free (p. 776)	Frees a buffer allocated by the Virtuozzo API library and returned to the client.

PrlBuffer_Free Function

Frees a buffer allocated by the Virtuozzo API library and returned to the client.

Syntax

```
PRL_RESULT PrlBuffer_Free(
    PRL_VOID_PTR buffer
);
```

File

PrlApiCore.h

Parameters

buffer

A pointer to a buffer to free.

Returns

PRL_RESULT. Possible values:

PRL_ERR_INVALID_ARG (p. 1290) - invalid pointer to the buffer was specified.

PRL_ERR_SUCCESS (p. 1365) - operation completed successfully.

Remarks

Every time a pointer to a buffer containing the data is returned to the client, it must be freed using this function. Failure to do so will result in memory leaks.

Remote Desktop Access

This chapter contains information about functions that can be used to remotely capture screen shots of a virtual machine desktop and to send keyboard and mouse commands to it.

Description

With this functionality, you can programmatically connect to a remote virtual machine and run its applications as if you were sitting at the virtual machine's console. The typical uses of the Remote Desktop Access functionality are:

1. Creating automation scripts for unattended operating system and other software installations.
2. Creating your own remote desktop applications with graphical user interfaces.

There are four groups of functions comprising the Remote Desktop Access API :

1. Primary display capture. These functions allow to capture a region or an entire screen of the remote virtual machine. In scripts, you can take a snapshot of a particular screen in advance and then compare it to the screen captured at runtime. If comparison operation determines that you've got the screen of interest, you can interact with it by sending mouse or keyboard commands (pressing buttons, making selections, etc.). In a GUI application, you can constantly capture screens (or screen regions) and display them to the user of your application, thus providing a real-time remote desktop access to a virtual machine.
2. Secondary display capture. This group of functions is almost identical to the Primary Display Capture group with the exception that it operates on a secondary display(s) if one is installed in a virtual machine.
3. Mouse control. These functions provide control over the mouse in a virtual machine. You can change the position of the mouse pointer, press and release mouse buttons, and use a scroll wheel. In GUI applications, you can capture mouse events in the host OS where the application is running and then transmit those events to the virtual machine. In scripts, you can move a mouse pointer to the coordinates you know in advance (coordinates of a button, an edit control, etc.) and perform an action.
4. Keyboard control. These functions simply send a key/action code combination to the virtual machine. In scripts, you can use these functions to interact with controls on a window opened inside a virtual machine (pressing buttons, selecting options, etc.). In a typical GUI application, essential visual controls usually have keyboard shortcuts (accelerator keys) assigned to them. If a control doesn't have a shortcut, then you will have to use mouse control functions to position a mouse pointer over the control to select it.

Every remote desktop access session must begin with PrlDevDisplay_ConnectToVm function call and end with the PrlDevDisplay_DisconnectFromVm call.

For more information and code samples, please see **Virtuozzo Virtualization SDK Programmer's Guide**.

Topics

Name	Description
Audio Playback (p. 778)	Functions in this group provide functionality for remote sound playback.
Keyboard Control (p. 778)	Functions in this group provide functionality for sending keyboard commands to a remote virtual machine.
Mouse Control (p. 778)	Functions in this group provide functionality for sending mouse commands to a remote virtual machine.
Primary Display Capture (p. 778)	Functions in this group provide functionality for capturing screen shots of a remote virtual machine desktop.
Secondary Display Capture (p. 778)	Functions in this group provide functionality for capturing screen shots of a secondary display if one is installed in a virtual machine.

Audio Playback

Functions in this group provide functionality for remote sound playback.

Keyboard Control

Functions in this group provide functionality for sending keyboard commands to a remote virtual machine.

Mouse Control

Functions in this group provide functionality for sending mouse commands to a remote virtual machine.

Primary Display Capture

Functions in this group provide functionality for capturing screen shots of a remote virtual machine desktop.

Secondary Display Capture

Functions in this group provide functionality for capturing screen shots of a secondary display if one is installed in a virtual machine.

Callback Prototypes

This chapter describes callback function prototypes that are used to implement your own callback functions for use in asynchronous calls.

Topics

Name	Description
PRL_EVENT_HANDLER_PTR (p. 780)	PRL_EVENT_HANDLER_PTR is a function prototype for implementing callback functions for use in asynchronous calls.

Types

Name	Description
PRL_DISK_CALLBACK (p. 779)	Define type of user's callback function PRL_DISK_CALLBACK is a function prototype for implementing callback functions for use in PHT_VIRTUAL_DISK (p. 432) calls.
PRL_STATES_CALLBACK (p. 780)	Later put it somwhere else PRL_STATES_CALLBACK is a function prototype for implementing callback functions for use in PHT_VIRTUAL_DISK (p. 432) calls.

PRL_DISK_CALLBACK Type

Define type of user's callback function

PRL_DISK_CALLBACK is a function prototype for implementing callback functions for use in PHT_VIRTUAL_DISK (p. 432) calls.

Syntax

```
typedef PRL_BOOL (PRL_CALL *PRL_DISK_CALLBACK)(PRL_INT32 iComplete, PRL_INT32 iTotal,
PRL_VOID_PTR pParam);
```

File

PrlDisk.h

Parameters

iComplete

An integer value indicating the state of the disk operation completion. Values range from 0 to the value contained in the iTotal parameter. A value greater than 0 and less than iTotal indicates that the operation is still in progress and can be used to determine the state (percentage) of completion. A value greater than iTotal indicates that the operation completed successfully. A negative value indicates an error.

iTotal

An integer value indicating the upper bound of the disk operation completion range.

pParam

A pointer to a buffer containing user-defined data.

Remarks

A pointer to a callback function can be passed to another function that accepts it as a parameter. The operation initiated by that

function will be performed in the background thread, which will call the callback function (possibly more than once) passing to it intermediate results or the final result of the operation.

PRL_EVENT_HANDLER_PTR

PRL_EVENT_HANDLER_PTR is a function prototype for implementing callback functions for use in asynchronous calls.

Syntax

```
typedef PRL_METHOD_PTR( PRL_EVENT_HANDLER_PTR ) ( PRL_HANDLE hEvent, PRL_VOID_PTR data );
```

Parameters

hEvent

A handle of type PHT_EVENT (p. 72) or PHT_JOB (p. 138). The PHT_EVENT (p. 72) type indicates that the callback function was called by a system event. If the type is PHT_JOB (p. 138) then the callback function was called by an asynchronous job. The type can be determined using the PrlHandle_GetType (p. 750) function. For more information and code samples see **Virtuozzo Virtualization SDK Programmer's Guide**.

Note: You must always free the handle before exiting from the function regardless of whether you actually used it or not. Failure to do so will result in a memory leak.

data

A pointer to a buffer containing user-defined data.

Remarks

A pointer to a callback function can be passed to another function that accepts it as a parameter. The operation initiated by that function will be performed in the background thread, which will call the callback function (possibly more than once) passing to it intermediate results or the final result of the operation.

PRL_STATES_CALLBACK Type

Later put it somwhere else

PRL_STATES_CALLBACK is a function prototype for implementing callback functions for use in PHT_VIRTUAL_DISK (p. 432) calls.

Syntax

```
typedef PRL_BOOL (PRL_CALL *PRL_STATES_CALLBACK)(PRL_STATES_CALLBACK_TYPE iCallbackType,
PRL_INT32 iProgress, PRL_VOID_PTR pParameter);
```

File

PrlDisk.h

Parameters

iCallbackType

Callback type. At the time of this writing the only supported callback type is PRL_STATE_DISK_CALLBACK (p. 1563).

iProgress

An integer value indicating the state of the disk operation completion. To determine the state (e.g. percentage of completion), compare the received value to the following constants:

PRL_STATE_PROGRESS_MIN (p. 1563) -- The lower bound of the operation completion range.

PRL_STATE_PROGRESS_MAX (p. 1563) -- The upper bound of the operation completion range.

PRL_STATE_PROGRESS_COMPLETED (p. 1563) -- Indicates that the operation completed successfully.

A value between PRL_STATE_PROGRESS_MIN (p. 1563) and PRL_STATE_PROGRESS_MAX (p. 1563) indicates that the operation is still in progress. A value greater than PRL_STATE_PROGRESS_COMPLETED (p. 1563) indicates that the operation completed successfully. A negative value indicates an error.

pParameter

A pointer to a buffer containing user-defined data.

Remarks

A pointer to a callback function can be passed to another function that accepts it as a parameter. The operation initiated by that function will be performed in the background thread, which will call the callback function (possibly more than once) passing to it intermediate results or the final result of the operation.

Types

Topics

Name	Description
PRL_BOOL Types (p. 782)	
PRL_CHAR Types (p. 783)	
PRL_CONST Types (p. 784)	
PRL_DEVICE Types (p. 788)	
PRL_DISK Types (p. 788)	
PRL_DOUBLE Types (p. 789)	
PRL_FILE Types (p. 790)	
PRL_INT Types (p. 791)	
PRL_IO Types (p. 794)	
PRL_TIS Types (p. 796)	
PRL_TOOLS Types (p. 796)	
PRL_UINT Types (p. 797)	
PRL_VM Types (p. 800)	
PRL_VOID Types (p. 808)	
Other PRL Types (p. 809)	
Other Types (p. 835)	

PRL_BOOL Types

Types

Name	Description
PRL_BOOL (p. 782)	Boolean value (32 bit integer), zero means FALSE or TRUE otherwise
PRL_BOOL_PTR (p. 783)	

PRL_BOOL Type

Boolean value (32 bit integer), zero means FALSE or TRUE otherwise

Syntax

```
typedef unsigned int PRL_BOOL;
```

File

PrlTypes.h

PRL_BOOL_PTR Type

Syntax

```
typedef PRL_BOOL* PRL_BOOL_PTR;
```

File

PrlTypes.h

PRL_CHAR Types

Types

Name	Description
PRL_CHAR (p. 783)	All strings are UTF8 encoded
PRL_CHAR_PTR (p. 783)	

PRL_CHAR Type

All strings are UTF8 encoded

Syntax

```
typedef char PRL_CHAR;
```

File

PrlTypes.h

PRL_CHAR_PTR Type

Syntax

```
typedef PRL_CHAR* PRL_CHAR_PTR;
```

File

PrlTypes.h

PRL_CONST Types

Types

Name	Description
PRL_CONST_BOOL_PTR (p. 784)	
PRL_CONST_DISK_OPEN_FLAGS_PTR (p. 784)	
PRL_CONST_DISK_PARAMETERS_PTR (p. 785)	
PRL_CONST_DOUBLE_PTR (p. 785)	
PRL_CONST_HANDLE_PTR (p. 785)	
PRL_CONST_INT16_PTR (p. 785)	
PRL_CONST_INT32_PTR (p. 785)	
PRL_CONST_INT64_PTR (p. 786)	
PRL_CONST_INT8_PTR (p. 786)	
PRL_CONST_SIZE_PTR (p. 786)	
PRL_CONST_SNAPSHOT_ELEMENT_PTR (p. 786)	
PRL_CONST_STR (p. 786)	
PRL_CONST_STR_PTR (p. 787)	
PRL_CONST_UINT16_PTR (p. 787)	
PRL_CONST_UINT32_PTR (p. 787)	
PRL_CONST_UINT64_PTR (p. 787)	
PRL_CONST_UINT8_PTR (p. 787)	
PRL_CONST_VOID_PTR (p. 788)	

PRL_CONST_BOOL_PTR Type

Syntax

```
typedef const PRL_BOOL* PRL_CONST_BOOL_PTR;
```

File

PrlTypes.h

PRL_CONST_DISK_OPEN_FLAGS_PTR Type

Syntax

```
typedef const PRL_DISK_OPEN_FLAGS* PRL_CONST_DISK_OPEN_FLAGS_PTR;
```

File

PrlDisk.h

PRL_CONST_DISK_PARAMETERS_PTR Type

Syntax

```
typedef const PRL_DISK_PARAMETERS* PRL_CONST_DISK_PARAMETERS_PTR;
```

File

PrlDisk.h

PRL_CONST_DOUBLE_PTR Type

Syntax

```
typedef const PRL_DOUBLE* PRL_CONST_DOUBLE_PTR;
```

File

PrlTypes.h

PRL_CONST_HANDLE_PTR Type

Syntax

```
typedef const PRL_HANDLE* PRL_CONST_HANDLE_PTR;
```

File

PrlTypes.h

PRL_CONST_INT16_PTR Type

Syntax

```
typedef const PRL_INT16* PRL_CONST_INT16_PTR;
```

File

PrlTypes.h

PRL_CONST_INT32_PTR Type

Syntax

```
typedef const PRL_INT32* PRL_CONST_INT32_PTR;
```

File

PrlTypes.h

PRL_CONST_INT64_PTR Type

Syntax

```
typedef const PRL_INT64* PRL_CONST_INT64_PTR;
```

File

PrlTypes.h

PRL_CONST_INT8_PTR Type

Syntax

```
typedef const PRL_INT8* PRL_CONST_INT8_PTR;
```

File

PrlTypes.h

PRL_CONST_SIZE_PTR Type

Syntax

```
typedef const PRL_SIZE* PRL_CONST_SIZE_PTR;
```

File

PrlTypes.h

PRL_CONST_SNAPSHOT_ELEMENT_PTR Type

Syntax

```
typedef const PRL_SNAPSHOT_ELEMENT* PRL_CONST_SNAPSHOT_ELEMENT_PTR;
```

File

PrlDisk.h

PRL_CONST_STR Type

Syntax

```
typedef const char* PRL_CONST_STR;
```

File

PrlTypes.h

PRL_CONST_STR_PTR Type

Syntax

```
typedef PRL_CONST_STR* PRL_CONST_STR_PTR;
```

File

PrlTypes.h

PRL_CONST_UINT16_PTR Type

Syntax

```
typedef const PRL_UINT16* PRL_CONST_UINT16_PTR;
```

File

PrlTypes.h

PRL_CONST_UINT32_PTR Type

Syntax

```
typedef const PRL_UINT32* PRL_CONST_UINT32_PTR;
```

File

PrlTypes.h

PRL_CONST_UINT64_PTR Type

Syntax

```
typedef const PRL_UINT64* PRL_CONST_UINT64_PTR;
```

File

PrlTypes.h

PRL_CONST_UINT8_PTR Type

Syntax

```
typedef const PRL_UINT8* PRL_CONST_UINT8_PTR;
```

File

PrlTypes.h

PRL_CONST_VOID_PTR Type

Syntax

```
typedef const PRL_VOID* PRL_CONST_VOID_PTR;
```

File

PrlTypes.h

PRL_DEVICE Types

Types

Name	Description
PRL_DEVICE_ACTION_PTR (p. 788)	
PRL_DEVICE_TYPE_PTR (p. 788)	

PRL_DEVICE_ACTION_PTR Type

Syntax

```
typedef PRL_DEVICE_ACTION* PRL_DEVICE_ACTION_PTR;
```

File

PrlEnums.h

PRL_DEVICE_TYPE_PTR Type

Syntax

```
typedef PRL_DEVICE_TYPE* PRL_DEVICE_TYPE_PTR;
```

File

PrlEnums.h

PRL_DISK Types

Types

Name	Description
PRL_DISK_OPEN_FLAGS (p. 789)	Storage/image/disk open parameters

PRL_DISK_OPEN_FLAGS_PTR (p. 789)	
PRL_DISK_PARAMETERS_PTR (p. 789)	

PRL_DISK_OPEN_FLAGS Type

Storage/image/disk open parameters

Syntax

```
typedef PRL_UINT32 PRL_DISK_OPEN_FLAGS;
```

File

PrlDisk.h

PRL_DISK_OPEN_FLAGS_PTR Type

Syntax

```
typedef PRL_DISK_OPEN_FLAGS* PRL_DISK_OPEN_FLAGS_PTR;
```

File

PrlDisk.h

PRL_DISK_PARAMETERS_PTR Type

Syntax

```
typedef PRL_DISK_PARAMETERS* PRL_DISK_PARAMETERS_PTR;
```

File

PrlDisk.h

PRL_DOUBLE Types

Types

Name	Description
PRL_DOUBLE (p. 790)	Double type
PRL_DOUBLE_PTR (p. 790)	
PRL_DOUBLE_PTR_PTR (p. 790)	

PRL_DOUBLE Type

Double type

Syntax

```
typedef double PRL_DOUBLE;
```

File

PrlTypes.h

PRL_DOUBLE_PTR Type

Syntax

```
typedef PRL_DOUBLE* PRL_DOUBLE_PTR;
```

File

PrlTypes.h

PRL_DOUBLE_PTR_PTR Type

Syntax

```
typedef PRL_DOUBLE_PTR* PRL_DOUBLE_PTR_PTR;
```

File

PrlTypes.h

PRL_FILE Types

Types

Name	Description
PRL_FILE_SYSTEM_ELEMENT_TYPE_PTR (p. 791)	
PRL_FILE_SYSTEM_FS_TYPE_PTR (p. 791)	
PRL_FILE_SYSTEM_TYPE_PTR (p. 791)	

PRL_FILE_SYSTEM_ELEMENT_TYPE_PTR Type

Syntax

```
typedef PRL_FILE_SYSTEM_ELEMENT_TYPE* PRL_FILE_SYSTEM_ELEMENT_TYPE_PTR;
```

File

PrlEnums.h

PRL_FILE_SYSTEM_FS_TYPE_PTR Type

Syntax

```
typedef PRL_FILE_SYSTEM_FS_TYPE* PRL_FILE_SYSTEM_FS_TYPE_PTR;
```

File

PrlEnums.h

PRL_FILE_SYSTEM_TYPE_PTR Type

Syntax

```
typedef PRL_FILE_SYSTEM_TYPE* PRL_FILE_SYSTEM_TYPE_PTR;
```

File

PrlEnums.h

PRL_INT Types

Types

Name	Description
PRL_INT16 (p. 792)	Signed 16 bit integer value
PRL_INT16_PTR (p. 792)	
PRL_INT16_PTR_PTR (p. 792)	
PRL_INT32 (p. 792)	Signed 32 bit integer value
PRL_INT32_PTR (p. 793)	
PRL_INT32_PTR_PTR (p. 793)	
PRL_INT64 (p. 793)	Signed 64 bit integer value Use _MSC_VER macros for MinGW use
PRL_INT64_PTR (p. 793)	
PRL_INT64_PTR_PTR (p. 793)	
PRL_INT8 (p. 794)	Signed 8 bit integer value

PRL_INT8_PTR (p. 794)	
PRL_INT8_PTR_PTR (p. 794)	

PRL_INT16 Type

Signed 16 bit integer value

Syntax

```
typedef short PRL_INT16;
```

File

PrlTypes.h

PRL_INT16_PTR Type

Syntax

```
typedef PRL_INT16* PRL_INT16_PTR;
```

File

PrlTypes.h

PRL_INT16_PTR_PTR Type

Syntax

```
typedef PRL_INT16** PRL_INT16_PTR_PTR;
```

File

PrlTypes.h

PRL_INT32 Type

Signed 32 bit integer value

Syntax

```
typedef int PRL_INT32;
```

File

PrlTypes.h

PRL_INT32_PTR Type

Syntax

```
typedef PRL_INT32* PRL_INT32_PTR;
```

File

PrlTypes.h

PRL_INT32_PTR_PTR Type

Syntax

```
typedef PRL_INT32** PRL_INT32_PTR_PTR;
```

File

PrlTypes.h

PRL_INT64 Type

Signed 64 bit integer value Use _MSC_VER macros for MinGW use

Syntax

```
typedef long long PRL_INT64;
```

File

PrlTypes.h

PRL_INT64_PTR Type

Syntax

```
typedef PRL_INT64* PRL_INT64_PTR;
```

File

PrlTypes.h

PRL_INT64_PTR_PTR Type

Syntax

```
typedef PRL_INT64** PRL_INT64_PTR_PTR;
```

File

PrlTypes.h

PRL_INT8 Type

Signed 8 bit integer value

Syntax

```
typedef signed char PRL_INT8;
```

File

PrlTypes.h

PRL_INT8_PTR Type

Syntax

```
typedef PRL_INT8* PRL_INT8_PTR;
```

File

PrlTypes.h

PRL_INT8_PTR_PTR Type

Syntax

```
typedef PRL_INT8** PRL_INT8_PTR_PTR;
```

File

PrlTypes.h

PRL_IO Types

Types

Name	Description
PRL_IO_DISCONNECT_REASON_PTR (p. 795)	
PRL_IO_DISPLAY_SCREEN_SIZE_PTR (p. 795)	
PRL_IO_PALETTE (p. 795)	
PRL_IO_SCREEN_SIZE_PTR (p. 795)	
PRL_IO_STATE_PTR (p. 795)	

PRL_IO_DISCONNECT_REASON_PTR Type

Syntax

```
typedef PRL_IO_DISCONNECT_REASON* PRL_IO_DISCONNECT_REASON_PTR;
```

File

PrlEnums.h

PRL_IO_DISPLAY_SCREEN_SIZE_PTR Type

Syntax

```
typedef PRL_IO_DISPLAY_SCREEN_SIZE* PRL_IO_DISPLAY_SCREEN_SIZE_PTR;
```

File

PrlIOStructs.h

PRL_IO_PALETTE Type

Syntax

```
typedef PRL_UINT32 PRL_IO_PALETTE[256];
```

File

PrlIOStructs.h

PRL_IO_SCREEN_SIZE_PTR Type

Syntax

```
typedef PRL_IO_SCREEN_SIZE* PRL_IO_SCREEN_SIZE_PTR;
```

File

PrlIOStructs.h

PRL_IO_STATE_PTR Type

Syntax

```
typedef PRL_IO_STATE* PRL_IO_STATE_PTR;
```

File

PrlEnums.h

PRL_TIS Types

Types

Name	Description
PRL_TIS_RECORD_FIELD_PTR (p. 796)	
PRL_TIS_RECORD_STATE_PTR (p. 796)	

PRL_TIS_RECORD_FIELD_PTR Type

Syntax

```
typedef PRL_TIS_RECORD_FIELD* PRL_TIS_RECORD_FIELD_PTR;
```

File

PrlEnums.h

PRL_TIS_RECORD_STATE_PTR Type

Syntax

```
typedef PRL_TIS_RECORD_STATE* PRL_TIS_RECORD_STATE_PTR;
```

File

PrlEnums.h

PRL_TOOLS Types

Types

Name	Description
PRL_TOOLS_SIA_OPEN_IN_TYPES_PTR (p. 797)	
PRL_TOOLS_UTILITY_COMMAND_FLAG_PTR (p. 797)	
PRL_TOOLS_UTILITY_COMMAND_ID_PTR (p. 797)	

PRL_TOOLS_SIA_OPEN_IN_TYPES_PTR Type

Syntax

```
typedef PRL_TOOLS_SIA_OPEN_IN_TYPES* PRL_TOOLS_SIA_OPEN_IN_TYPES_PTR;
```

File

PrlEnums.h

PRL_TOOLS_UTILITY_COMMAND_FLAG_PTR Type

Syntax

```
typedef PRL_TOOLS_UTILITY_COMMAND_FLAG* PRL_TOOLS_UTILITY_COMMAND_FLAG_PTR;
```

File

PrlCommandsFlags.h

PRL_TOOLS_UTILITY_COMMAND_ID_PTR Type

Syntax

```
typedef PRL_TOOLS_UTILITY_COMMAND_ID* PRL_TOOLS_UTILITY_COMMAND_ID_PTR;
```

File

PrlEnums.h

PRL_UINT Types

Types

Name	Description
PRL_UINT16 (p. 798)	Unsigned 16 bit integer value
PRL_UINT16_PTR (p. 798)	
PRL_UINT16_PTR_PTR (p. 798)	
PRL_UINT32 (p. 798)	Unsigned 32 bit integer value
PRL_UINT32_PTR (p. 799)	
PRL_UINT32_PTR_PTR (p. 799)	
PRL_UINT64 (p. 799)	Unsigned 64 bit integer value Use _MSC_VER macros for MinGW use
PRL_UINT64_PTR (p. 799)	
PRL_UINT64_PTR_PTR (p. 799)	
PRL_UINT8 (p. 800)	Unsigned 8 bit integer value

PRL_UINT8_PTR (p. 800)	
PRL_UINT8_PTR_PTR (p. 800)	

PRL_UINT16 Type

Unsigned 16 bit integer value

Syntax

```
typedef unsigned short PRL_UINT16;
```

File

PrlTypes.h

PRL_UINT16_PTR Type

Syntax

```
typedef PRL_UINT16* PRL_UINT16_PTR;
```

File

PrlTypes.h

PRL_UINT16_PTR_PTR Type

Syntax

```
typedef PRL_UINT16** PRL_UINT16_PTR_PTR;
```

File

PrlTypes.h

PRL_UINT32 Type

Unsigned 32 bit integer value

Syntax

```
typedef unsigned int PRL_UINT32;
```

File

PrlTypes.h

PRL_UINT32_PTR Type

Syntax

```
typedef PRL_UINT32* PRL_UINT32_PTR;
```

File

PrlTypes.h

PRL_UINT32_PTR_PTR Type

Syntax

```
typedef PRL_UINT32** PRL_UINT32_PTR_PTR;
```

File

PrlTypes.h

PRL_UINT64 Type

Unsigned 64 bit integer value Use _MSC_VER macros for MinGW use

Syntax

```
typedef unsigned long long PRL_UINT64;
```

File

PrlTypes.h

PRL_UINT64_PTR Type

Syntax

```
typedef PRL_UINT64* PRL_UINT64_PTR;
```

File

PrlTypes.h

PRL_UINT64_PTR_PTR Type

Syntax

```
typedef PRL_UINT64** PRL_UINT64_PTR_PTR;
```

File

PrlTypes.h

PRL_UINT8 Type

Unsigned 8 bit integer value

Syntax

```
typedef unsigned char PRL_UINT8;
```

File

PrlTypes.h

PRL_UINT8_PTR Type

Syntax

```
typedef PRL_UINT8* PRL_UINT8_PTR;
```

File

PrlTypes.h

PRL_UINT8_PTR_PTR Type

Syntax

```
typedef PRL_UINT8** PRL_UINT8_PTR_PTR;
```

File

PrlTypes.h

PRL_VM Types

Types

Name	Description
PRL_VM_ACCELERATION_LEVEL_PTR (p. 801)	
PRL_VM_ACCESS_FOR_OTHERS_PTR (p. 802)	
PRL_VM_ACTION_ON_STOP_PTR (p. 802)	
PRL_VM_ACTION_ON_WINDOW_CLOSE_PTR (p. 802)	

PRL_VM_ACTION_PTR (p. 802)	
PRL_VM_AUTOSTART_OPTION_PTR (p. 802)	
PRL_VM_AUTOSTOP_OPTION_PTR (p. 803)	
PRL_VM_BACKUP_TYPE_PTR (p. 803)	
PRL_VM_BOOT_ORDER_PTR (p. 803)	
PRL_VM_COLOR_PTR (p. 803)	
PRL_VM_CONFIG_SECTIONS_PTR (p. 803)	
PRL_VM_DEV_EMULATION_TYPE_PTR (p. 804)	
PRL_VM_EXEC_APP_FILE_DESCRIPTOR_PTR (p. 804)	
PRL_VM_EXEC_RUN_PROGRAM_MODES_PTR (p. 804)	
PRL_VM_LOCATION_PTR (p. 804)	
PRL_VM_MOUSE_CURSOR_STATE_PTR (p. 804)	
PRL_VM_MOUSE_STATE_PTR (p. 805)	
PRL_VM_NET_ADAPTER_TYPE_PTR (p. 805)	
PRL_VM_NET_ADAPTER_WIFI_RULE_PTR (p. 805)	
PRL_VM_NORMAL_START_FLAGS_PTR (p. 805)	
PRL_VM_ON_CRASH_ACTION_PTR (p. 805)	
PRL_VM_ON_CRASH_OPTION_PTR (p. 806)	
PRL_VM_PRIORITY_PTR (p. 806)	
PRL_VM_REMOTE_DISPLAY_MODE_PTR (p. 806)	
PRL_VM_SMARTGUARD_SCHEMA_PTR (p. 806)	
PRL_VM_START_LOGIN_MODE_PTR (p. 806)	
PRL_VM_START_MODE_PTR (p. 807)	
PRL_VM_STATISTICS_FLAGS_PTR (p. 807)	
PRL_VM_TOOLS_STATE_PTR (p. 807)	
PRL_VM_TYPE_PTR (p. 807)	
PRL_VM_WINDOW_MODE_PTR (p. 807)	

PRL_VM_ACCELERATION_LEVEL_PTR Type

Syntax

```
typedef PRL_VM_ACCELERATION_LEVEL* PRL_VM_ACCELERATION_LEVEL_PTR;
```

File

PrlEnums.h

PRL_VM_ACCESS_FOR_OTHERS_PTR Type

Syntax

```
typedef PRL_VM_ACCESS_FOR_OTHERS* PRL_VM_ACCESS_FOR_OTHERS_PTR;
```

File

PrlEnums.h

PRL_VM_ACTION_ON_STOP_PTR Type

Syntax

```
typedef PRL_VM_ACTION_ON_STOP* PRL_VM_ACTION_ON_STOP_PTR;
```

File

PrlEnums.h

PRL_VM_ACTION_ON_WINDOW_CLOSE_PTR Type

Syntax

```
typedef PRL_VM_ACTION_ON_WINDOW_CLOSE* PRL_VM_ACTION_ON_WINDOW_CLOSE_PTR;
```

File

PrlEnums.h

PRL_VM_ACTION_PTR Type

Syntax

```
typedef PRL_VM_ACTION* PRL_VM_ACTION_PTR;
```

File

PrlEnums.h

PRL_VM_AUTOSTART_OPTION_PTR Type

Syntax

```
typedef PRL_VM_AUTOSTART_OPTION* PRL_VM_AUTOSTART_OPTION_PTR;
```

File

PrlEnums.h

PRL_VM_AUTOSTOP_OPTION_PTR Type

Syntax

```
typedef PRL_VM_AUTOSTOP_OPTION* PRL_VM_AUTOSTOP_OPTION_PTR;
```

File

PrlEnums.h

PRL_VM_BACKUP_TYPE_PTR Type

Syntax

```
typedef PRL_VM_BACKUP_TYPE* PRL_VM_BACKUP_TYPE_PTR;
```

File

PrlEnums.h

PRL_VM_BOOT_ORDER_PTR Type

Syntax

```
typedef PRL_VM_BOOT_ORDER* PRL_VM_BOOT_ORDER_PTR;
```

File

PrlEnums.h

PRL_VM_COLOR_PTR Type

Syntax

```
typedef PRL_VM_COLOR* PRL_VM_COLOR_PTR;
```

File

PrlEnums.h

PRL_VM_CONFIG_SECTIONS_PTR Type

Syntax

```
typedef PRL_VM_CONFIG_SECTIONS* PRL_VM_CONFIG_SECTIONS_PTR;
```

File

PrlEnums.h

PRL_VM_DEV_EMULATION_TYPE_PTR Type

Syntax

```
typedef PRL_VM_DEV_EMULATION_TYPE* PRL_VM_DEV_EMULATION_TYPE_PTR;
```

File

PrlEnums.h

PRL_VM_EXEC_APP_FILE_DESCRIPTOR_PTR Type

Syntax

```
typedef PRL_VM_EXEC_APP_FILE_DESCRIPTOR* PRL_VM_EXEC_APP_FILE_DESCRIPTOR_PTR;
```

File

PrlCommandsFlags.h

PRL_VM_EXEC_RUN_PROGRAM_MODES_PTR Type

Syntax

```
typedef PRL_VM_EXEC_RUN_PROGRAM_MODES* PRL_VM_EXEC_RUN_PROGRAM_MODES_PTR;
```

File

PrlCommandsFlags.h

PRL_VM_LOCATION_PTR Type

Syntax

```
typedef PRL_VM_LOCATION* PRL_VM_LOCATION_PTR;
```

File

PrlEnums.h

PRL_VM_MOUSE_CURSOR_STATE_PTR Type

Syntax

```
typedef PRL_VM_MOUSE_CURSOR_STATE* PRL_VM_MOUSE_CURSOR_STATE_PTR;
```

File

PrlEnums.h

PRL_VM_MOUSE_STATE_PTR Type

Syntax

```
typedef PRL_VM_MOUSE_STATE* PRL_VM_MOUSE_STATE_PTR;
```

File

PrlEnums.h

PRL_VM_NET_ADAPTER_TYPE_PTR Type

Syntax

```
typedef PRL_VM_NET_ADAPTER_TYPE* PRL_VM_NET_ADAPTER_TYPE_PTR;
```

File

PrlEnums.h

PRL_VM_NET_ADAPTER_WIFI_RULE_PTR Type

Syntax

```
typedef PRL_VM_NET_ADAPTER_WIFI_RULE* PRL_VM_NET_ADAPTER_WIFI_RULE_PTR;
```

File

PrlEnums.h

PRL_VM_NORMAL_START_FLAGS_PTR Type

Syntax

```
typedef PRL_VM_NORMAL_START_FLAGS* PRL_VM_NORMAL_START_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_VM_ON_CRASH_ACTION_PTR Type

Syntax

```
typedef PRL_VM_ON_CRASH_ACTION* PRL_VM_ON_CRASH_ACTION_PTR;
```

File

PrlEnums.h

PRL_VM_ON_CRASH_OPTION_PTR Type

Syntax

```
typedef PRL_VM_ON_CRASH_OPTION* PRL_VM_ON_CRASH_OPTION_PTR;
```

File

PrlCommandsFlags.h

PRL_VM_PRIORITY_PTR Type

Syntax

```
typedef PRL_VM_PRIORITY* PRL_VM_PRIORITY_PTR;
```

File

PrlEnums.h

PRL_VM_REMOTE_DISPLAY_MODE_PTR Type

Syntax

```
typedef PRL_VM_REMOTE_DISPLAY_MODE* PRL_VM_REMOTE_DISPLAY_MODE_PTR;
```

File

PrlEnums.h

PRL_VM_SMARTGUARD_SCHEMA_PTR Type

Syntax

```
typedef PRL_VM_SMARTGUARD_SCHEMA* PRL_VM_SMARTGUARD_SCHEMA_PTR;
```

File

PrlEnums.h

PRL_VM_START_LOGIN_MODE_PTR Type

Syntax

```
typedef PRL_VM_START_LOGIN_MODE* PRL_VM_START_LOGIN_MODE_PTR;
```

File

PrlEnums.h

PRL_VM_START_MODE_PTR Type

Syntax

```
typedef PRL_VM_START_MODE* PRL_VM_START_MODE_PTR;
```

File

PrlCommandsFlags.h

PRL_VM_STATISTICS_FLAGS_PTR Type

Syntax

```
typedef PRL_VM_STATISTICS_FLAGS* PRL_VM_STATISTICS_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_VM_TOOLS_STATE_PTR Type

Syntax

```
typedef PRL_VM_TOOLS_STATE* PRL_VM_TOOLS_STATE_PTR;
```

File

PrlEnums.h

PRL_VM_TYPE_PTR Type

Syntax

```
typedef PRL_VM_TYPE* PRL_VM_TYPE_PTR;
```

File

PrlEnums.h

PRL_VM_WINDOW_MODE_PTR Type

Syntax

```
typedef PRL_VM_WINDOW_MODE* PRL_VM_WINDOW_MODE_PTR;
```

File

PrlEnums.h

PRL_VOID Types

Types

Name	Description
PRL_VOID (p. 808)	Void pointers (for unified naming)
PRL_VOID_PTR (p. 808)	
PRL_VOID_PTR_PTR (p. 808)	

PRL_VOID Type

Void pointers (for unified naming)

Syntax

```
typedef void PRL_VOID;
```

File

PrlTypes.h

PRL_VOID_PTR Type

Syntax

```
typedef PRL_VOID* PRL_VOID_PTR;
```

File

PrlTypes.h

PRL_VOID_PTR_PTR Type

Syntax

```
typedef PRL_VOID_PTR* PRL_VOID_PTR_PTR;
```

File

PrlTypes.h

Other PRL Types

Types

Name	Description
PRL_ALLOWED_VM_COMMAND_PTR (p. 812)	
PRL_API_COMMAND_FLAGS_PTR (p. 812)	
PRL_API_INIT_FLAGS_PTR (p. 812)	
PRL_APP_IN_DOCK_MODE_PTR (p. 812)	
PRL_APPLIANCE_DOWNLOAD_STATUS_PTR (p. 812)	
PRL_APPLICATION_MODE_PTR (p. 813)	
PRL_BIOS_TYPE_PTR (p. 813)	
PRL_CHS_PTR (p. 813)	
PRL_CLONE_VM_FLAGS_PTR (p. 813)	
PRL_CLUSTERED_DEVICE_SUBTYPE_PTR (p. 814)	
PRL_COMMIT_ENCRYPTION_FLAGS_PTR (p. 814)	
PRL_COMPACT_MODE_PTR (p. 814)	
PRL_CORNER_ACTIONS_PTR (p. 814)	
PRL_CPU_HVT_PTR (p. 814)	
PRL_CPU_MODE_PTR (p. 815)	
PRL_CPU_VTX_MODE (p. 815)	
PRL_CPU_VTX_MODE_PTR (p. 815)	
PRL_CREATE_DISK_IMAGE_FLAGS_PTR (p. 815)	
PRL_CT_CAPMASK_PTR (p. 815)	
PRL_CT_REINSTALL_OPTIONS_PTR (p. 816)	
PRL_CT_TEMPLATE_TYPE_PTR (p. 816)	
PRL_DATA_STATISTIC_SEGMENTS_PTR (p. 816)	
PRL_DISPLAY_BUFFER_FORMAT_PTR (p. 816)	
PRL_DISPLAY_CONFIG_MODE_PTR (p. 816)	
PRL_DISTRO_MEDIA_TYPE_PTR (p. 817)	
PRL_DOCK_ICON_TYPE_PTR (p. 817)	
PRL_EVENT_ISSUER_TYPE_PTR (p. 817)	
PRL_EVENT_TYPE_PTR (p. 817)	
PRL_EXTERNAL_VM_TYPE_PTR (p. 817)	
PRL_FEATURES_MATRIX_PTR (p. 818)	
PRL_FIREWALL_DIRECTION_PTR (p. 818)	
PRL_FIREWALL_POLICY_PTR (p. 818)	
PRL_GENERIC_DEVICE_STATE_PTR (p. 818)	

PRL_GENERIC_PCI_DEVICE_CLASS_PTR (p. 818)	
PRL_GET_VM_CONFIG_FLAGS_PTR (p. 819)	
PRL_GET_VM_LIST_FLAGS_PTR (p. 819)	
PRL_GRACEFUL_SHUTDOWN_CAPABILITIES_PTR (p. 819)	
PRL_GRACEFUL_SHUTDOWN_KIND_PTR (p. 819)	
PRL_GUEST_DISTRO_VERSION_ENUM_PTR (p. 819)	
PRL_GUEST_OS_SUPPORT_TYPE_PTR (p. 820)	
PRL_HANDLE (p. 820)	Handle representation
PRL_HANDLE_PTR (p. 820)	
PRL_HANDLE_TYPE_PTR (p. 820)	
PRL_HARD_DISK_IFACE_MASK_PTR (p. 820)	
PRL_HARD_DISK_INTERNAL_FORMAT_PTR (p. 821)	
PRL_HOST_OS_TYPE_PTR (p. 821)	
PRL_HW_INFO_NET_ADAPTER_TYPE_PTR (p. 821)	
PRL_IMAGE_FORMAT_PTR (p. 821)	
PRL_IMAGE_PARAMETERS_PTR (p. 822)	
PRL_IMAGE_TYPE (p. 822)	Virtuozzo images types
PRL_IMAGE_TYPE_PTR (p. 822)	
PRL_JOB_OPERATION_CODE_PTR (p. 822)	
PRL_JOB_STATUS_PTR (p. 822)	
PRL_KEY_EVENT_PTR (p. 823)	
PRL_LICENSE_RESTRICTION_KEY_PTR (p. 823)	
PRL_LOGIN_LOCAL_FLAGS_PTR (p. 823)	
PRL_MASS_STORAGE_INTERFACE_TYPE_PTR (p. 823)	
PRL_MESSAGE_TYPE_ENUM_PTR (p. 824)	
PRL_MOBILE_ADVANCED_AUTH_MODE_PTR (p. 824)	
PRL_MOUSE_BUTTON_PTR (p. 824)	
PRL_NET_ADAPTER_EMULATED_TYPE_PTR (p. 824)	
PRL_NET_ADAPTER_TYPE_PTR (p. 824)	
PRL_NET_VIRTUAL_NETWORK_TYPE_PTR (p. 825)	
PRL_NETFILTER_MODE_PTR (p. 825)	
PRL_NETWORK_RATE_MPU_PTR (p. 825)	
PRL_ONLINE_COMPACT_MODE_PTR (p. 825)	
PRL_OPTIMIZE_MODIFIERS_MODE_PTR (p. 825)	
PRL_PARAM_FIELD_DATA_TYPE_PTR (p. 826)	
PRL_PORT_FORWARDING_TYPE_PTR (p. 826)	
PRL_PRINTER_INTERFACE_TYPE_PTR (p. 826)	
PRL_PRIVATE_EVENT_TYPE_PTR (p. 826)	

PRL_PROBLEM_REPORT_FLAGS_PTR (p. 826)	
PRL_PROBLEM_REPORT_REASON_PTR (p. 827)	
PRL_PROBLEM_REPORT_SCHEME_PTR (p. 827)	
PRL_PROBLEM_REPORT_TYPE_PTR (p. 827)	
PRL_PROCESS_STATE_TYPE_PTR (p. 827)	
PRL_PRODUCT_ENUM_PTR (p. 827)	
PRL_PROXY_CONNECTION_STATUS_PTR (p. 828)	
PRL_QUERY_ELEMENT_FLAGS_PTR (p. 828)	
PRL_REGISTER_3PARTY_VM_FLAGS_PTR (p. 828)	
PRL_REGISTER_NEW_VM_PTR (p. 828)	
PRL_REGISTER_VM_FLAGS_PTR (p. 828)	
PRL_RESIZE_IMAGE_FLAGS_PTR (p. 829)	
PRL_RUNNING_TASK_TYPE_ENUM_PTR (p. 829)	
PRL_SCALE_VIEW_MODE_PTR (p. 829)	
PRL_SECURITY_LEVEL_PTR (p. 829)	
PRL_SERIAL_PORT_SOCKET_OPERATION_MODE_PTR (p. 829)	
PRL_SERVICE_STATUS_ENUM_PTR (p. 830)	
PRL_SET_USERPASSWD_FLAGS_PTR (p. 830)	
PRL_SIZE (p. 830)	Unsigned integer type of the result of the sizeof operator
PRL_SIZE_PTR (p. 830)	
PRL_SIZE_PTR_PTR (p. 831)	
PRL_SNAPSHOT_ELEMENT_PTR (p. 831)	
PRL_SOUND_CHANNEL_TYPE_PTR (p. 831)	
PRL_SRV_SHUTDOWN_FLAGS_PTR (p. 831)	
PRL_STATES_CALLBACK_TYPE (p. 831)	Saved states callback types @author antonz@
PRL_STATES_CALLBACK_TYPE_PTR (p. 832)	
PRL_STR (p. 832)	
PRL_UNDO_DISKS_MODE_PTR (p. 832)	
PRL_UPDATE_LICENSE_FLAGS_PTR (p. 832)	
PRL_USB_DEVICE_AUTO_CONNECT_OPTION_PTR (p. 833)	
PRL_USB_DEVICE_CONNECT_REASON_PTR (p. 833)	
PRL_USB_DEVICE_TYPE_PTR (p. 833)	
PRL_VCMMD_COMMAND_FLAGS_PTR (p. 833)	
PRL_VERSION_ENUM_PTR (p. 833)	
PRL_VIDEO_3D_ACCELERATION_PTR (p. 834)	
PRL_VIDEO_RECEIVER_CONNECTION_STATE_PTR (p. 834)	
PRL_VIRTUAL_MACHINE_PROFILE_PTR (p. 834)	
PRL_VIRTUAL_NETWORK_FLAGS_PTR (p. 834)	
PRL_VMCTG_EVENT_PTR (p. 834)	
PRL_VTD_STATUS_CODE_PTR (p. 835)	
PRL_WINDOW_CORNERS_PTR (p. 835)	

PRL_WINDOW_EDGE_MASK_PTR (p. 835)	

PRL_ALLOWED_VM_COMMAND_PTR Type

Syntax

```
typedef PRL_ALLOWED_VM_COMMAND* PRL_ALLOWED_VM_COMMAND_PTR;
```

File

PrlEnums.h

PRL_API_COMMAND_FLAGS_PTR Type

Syntax

```
typedef PRL_API_COMMAND_FLAGS* PRL_API_COMMAND_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_API_INIT_FLAGS_PTR Type

Syntax

```
typedef PRL_API_INIT_FLAGS* PRL_API_INIT_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_APP_IN_DOCK_MODE_PTR Type

Syntax

```
typedef PRL_APP_IN_DOCK_MODE* PRL_APP_IN_DOCK_MODE_PTR;
```

File

PrlEnums.h

PRL_APPLIANCE_DOWNLOAD_STATUS_PTR Type

Syntax

```
typedef PRL_APPLIANCE_DOWNLOAD_STATUS* PRL_APPLIANCE_DOWNLOAD_STATUS_PTR;
```

File

PrlEnums.h

PRL_APPLICATION_MODE_PTR Type

Syntax

```
typedef PRL_APPLICATION_MODE* PRL_APPLICATION_MODE_PTR;
```

File

PrlEnums.h

PRL_BIOS_TYPE_PTR Type

Syntax

```
typedef PRL_BIOS_TYPE* PRL_BIOS_TYPE_PTR;
```

File

PrlEnums.h

PRL_CHS_PTR Type

Syntax

```
typedef PRL_CHS* PRL_CHS_PTR;
```

File

PrlDisk.h

PRL_CLONE_VM_FLAGS_PTR Type

Syntax

```
typedef PRL_CLONE_VM_FLAGS* PRL_CLONE_VM_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_CLUSTERED_DEVICE_SUBTYPE_PTR Type

Syntax

```
typedef PRL_CLUSTERED_DEVICE_SUBTYPE* PRL_CLUSTERED_DEVICE_SUBTYPE_PTR;
```

File

PrlEnums.h

PRL_COMMIT_ENCRYPTION_FLAGS_PTR Type

Syntax

```
typedef PRL_COMMIT_ENCRYPTION_FLAGS* PRL_COMMIT_ENCRYPTION_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_COMPACT_MODE_PTR Type

Syntax

```
typedef PRL_COMPACT_MODE* PRL_COMPACT_MODE_PTR;
```

File

PrlCommandsFlags.h

PRL_CORNER_ACTIONS_PTR Type

Syntax

```
typedef PRL_CORNER_ACTIONS* PRL_CORNER_ACTIONS_PTR;
```

File

PrlEnums.h

PRL_CPU_HVT_PTR Type

Syntax

```
typedef PRL_CPU_HVT* PRL_CPU_HVT_PTR;
```

File

PrlEnums.h

PRL_CPU_MODE_PTR Type

Syntax

```
typedef PRL_CPU_MODE* PRL_CPU_MODE_PTR;
```

File

PrlEnums.h

PRL_CPU_VTX_MODE Type

Syntax

```
typedef enum _PRL_CPU_HVT PRL_CPU_VTX_MODE;
```

File

PrlEnums.h

PRL_CPU_VTX_MODE_PTR Type

Syntax

```
typedef PRL_CPU_HVT* PRL_CPU_VTX_MODE_PTR;
```

File

PrlEnums.h

PRL_CREATE_DISK_IMAGE_FLAGS_PTR Type

Syntax

```
typedef PRL_CREATE_DISK_IMAGE_FLAGS* PRL_CREATE_DISK_IMAGE_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_CT_CAPMASK_PTR Type

Syntax

```
typedef PRL_CT_CAPMASK* PRL_CT_CAPMASK_PTR;
```

File

PrlEnums.h

PRL_CT_REINSTALL_OPTIONS_PTR Type

Syntax

```
typedef PRL_CT_REINSTALL_OPTIONS* PRL_CT_REINSTALL_OPTIONS_PTR;
```

File

PrlCommandsFlags.h

PRL_CT_TEMPLATE_TYPE_PTR Type

Syntax

```
typedef PRL_CT_TEMPLATE_TYPE* PRL_CT_TEMPLATE_TYPE_PTR;
```

File

PrlEnums.h

PRL_DATA_STATISTIC_SEGMENTS_PTR Type

Syntax

```
typedef PRL_DATA_STATISTIC_SEGMENTS* PRL_DATA_STATISTIC_SEGMENTS_PTR;
```

File

PrlEnums.h

PRL_DISPLAY_BUFFER_FORMAT_PTR Type

Syntax

```
typedef PRL_DISPLAY_BUFFER_FORMAT* PRL_DISPLAY_BUFFER_FORMAT_PTR;
```

File

PrlEnums.h

PRL_DISPLAY_CONFIG_MODE_PTR Type

Syntax

```
typedef PRL_DISPLAY_CONFIG_MODE* PRL_DISPLAY_CONFIG_MODE_PTR;
```

File

PrlEnums.h

PRL_DISTRO_MEDIA_TYPE_PTR Type

Syntax

```
typedef PRL_DISTRO_MEDIA_TYPE* PRL_DISTRO_MEDIA_TYPE_PTR;
```

File

PrlEnums.h

PRL_DOCK_ICON_TYPE_PTR Type

Syntax

```
typedef PRL_DOCK_ICON_TYPE* PRL_DOCK_ICON_TYPE_PTR;
```

File

PrlEnums.h

PRL_EVENT_ISSUER_TYPE_PTR Type

Syntax

```
typedef PRL_EVENT_ISSUER_TYPE* PRL_EVENT_ISSUER_TYPE_PTR;
```

File

PrlEnums.h

PRL_EVENT_TYPE_PTR Type

Syntax

```
typedef PRL_EVENT_TYPE* PRL_EVENT_TYPE_PTR;
```

File

PrlEventsValues.h

PRL_EXTERNAL_VM_TYPE_PTR Type

Syntax

```
typedef PRL_EXTERNAL_VM_TYPE* PRL_EXTERNAL_VM_TYPE_PTR;
```

File

PrlEnums.h

PRL_FEATURES_MATRIX_PTR Type

Syntax

```
typedef PRL_FEATURES_MATRIX* PRL_FEATURES_MATRIX_PTR;
```

File

PrlEnums.h

PRL_FIREWALL_DIRECTION_PTR Type

Syntax

```
typedef PRL_FIREWALL_DIRECTION* PRL_FIREWALL_DIRECTION_PTR;
```

File

PrlEnums.h

PRL_FIREWALL_POLICY_PTR Type

Syntax

```
typedef PRL_FIREWALL_POLICY* PRL_FIREWALL_POLICY_PTR;
```

File

PrlEnums.h

PRL_GENERIC_DEVICE_STATE_PTR Type

Syntax

```
typedef PRL_GENERIC_DEVICE_STATE* PRL_GENERIC_DEVICE_STATE_PTR;
```

File

PrlEnums.h

PRL_GENERIC_PCI_DEVICE_CLASS_PTR Type

Syntax

```
typedef PRL_GENERIC_PCI_DEVICE_CLASS* PRL_GENERIC_PCI_DEVICE_CLASS_PTR;
```

File

PrlEnums.h

PRL_GET_VM_CONFIG_FLAGS_PTR Type

Syntax

```
typedef PRL_GET_VM_CONFIG_FLAGS* PRL_GET_VM_CONFIG_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_GET_VM_LIST_FLAGS_PTR Type

Syntax

```
typedef PRL_GET_VM_LIST_FLAGS* PRL_GET_VM_LIST_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_GRACEFUL_SHUTDOWN_CAPABILITIES_PTR Type

Syntax

```
typedef PRL_GRACEFUL_SHUTDOWN_CAPABILITIES* PRL_GRACEFUL_SHUTDOWN_CAPABILITIES_PTR;
```

File

PrlEnums.h

PRL_GRACEFUL_SHUTDOWN_KIND_PTR Type

Syntax

```
typedef PRL_GRACEFUL_SHUTDOWN_KIND* PRL_GRACEFUL_SHUTDOWN_KIND_PTR;
```

File

PrlEnums.h

PRL_GUEST_DISTRO_VERSION_ENUM_PTR Type

Syntax

```
typedef PRL_GUEST_DISTRO_VERSION_ENUM* PRL_GUEST_DISTRO_VERSION_ENUM_PTR;
```

File

PrlEnums.h

PRL_GUEST_OS_SUPPORT_TYPE_PTR Type

Syntax

```
typedef PRL_GUEST_OS_SUPPORT_TYPE* PRL_GUEST_OS_SUPPORT_TYPE_PTR;
```

File

PrlEnums.h

PRL_HANDLE Type

Handle representation

Syntax

```
typedef int* PRL_HANDLE;
```

File

PrlTypes.h

PRL_HANDLE_PTR Type

Syntax

```
typedef PRL_HANDLE* PRL_HANDLE_PTR;
```

File

PrlTypes.h

PRL_HANDLE_TYPE_PTR Type

Syntax

```
typedef PRL_HANDLE_TYPE* PRL_HANDLE_TYPE_PTR;
```

File

PrlEnums.h

PRL_HARD_DISK_IFACE_MASK_PTR Type

Syntax

```
typedef PRL_HARD_DISK_IFACE_MASK* PRL_HARD_DISK_IFACE_MASK_PTR;
```

File

PrlEnums.h

PRL_HARD_DISK_INTERNAL_FORMAT_PTR Type

Syntax

```
typedef PRL_HARD_DISK_INTERNAL_FORMAT* PRL_HARD_DISK_INTERNAL_FORMAT_PTR;
```

File

PrlEnums.h

PRL_HOST_OS_TYPE_PTR Type

Syntax

```
typedef PRL_HOST_OS_TYPE* PRL_HOST_OS_TYPE_PTR;
```

File

PrlEnums.h

PRL_HW_INFO_NET_ADAPTER_TYPE_PTR Type

Syntax

```
typedef PRL_HW_INFO_NET_ADAPTER_TYPE* PRL_HW_INFO_NET_ADAPTER_TYPE_PTR;
```

File

PrlEnums.h

PRL_IMAGE_FORMAT_PTR Type

Syntax

```
typedef PRL_IMAGE_FORMAT* PRL_IMAGE_FORMAT_PTR;
```

File

PrlEnums.h

PRL_IMAGE_PARAMETERS_PTR Type

Syntax

```
typedef PRL_IMAGE_PARAMETERS* PRL_IMAGE_PARAMETERS_PTR;
```

File

PrlDisk.h

PRL_IMAGE_TYPE Type

Virtuozzo images types

Syntax

```
typedef PRL_UINT32 PRL_IMAGE_TYPE;
```

File

PrlDisk.h

PRL_IMAGE_TYPE_PTR Type

Syntax

```
typedef PRL_IMAGE_TYPE* PRL_IMAGE_TYPE_PTR;
```

File

PrlDisk.h

PRL_JOB_OPERATION_CODE_PTR Type

Syntax

```
typedef PRL_JOB_OPERATION_CODE* PRL_JOB_OPERATION_CODE_PTR;
```

File

PrlEnums.h

PRL_JOB_STATUS_PTR Type

Syntax

```
typedef PRL_JOB_STATUS* PRL_JOB_STATUS_PTR;
```

File

PrlEnums.h

PRL_KEY_EVENT_PTR Type

Syntax

```
typedef PRL_KEY_EVENT* PRL_KEY_EVENT_PTR;
```

File

PrlEnums.h

PRL_LICENSE_RESTRICTION_KEY_PTR Type

Syntax

```
typedef PRL_LICENSE_RESTRICTION_KEY* PRL_LICENSE_RESTRICTION_KEY_PTR;
```

File

PrlEnums.h

PRL_LOGIN_LOCAL_FLAGS_PTR Type

Syntax

```
typedef PRL_LOGIN_LOCAL_FLAGS* PRL_LOGIN_LOCAL_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_MASS_STORAGE_INTERFACE_TYPE_PTR Type

Syntax

```
typedef PRL_MASS_STORAGE_INTERFACE_TYPE* PRL_MASS_STORAGE_INTERFACE_TYPE_PTR;
```

File

PrlEnums.h

PRL_MESSAGE_TYPE_ENUM_PTR Type

Syntax

```
typedef PRL_MESSAGE_TYPE_ENUM* PRL_MESSAGE_TYPE_ENUM_PTR;
```

File

PrlEnums.h

PRL_MOBILE_ADVANCED_AUTH_MODE_PTR Type

Syntax

```
typedef PRL_MOBILE_ADVANCED_AUTH_MODE* PRL_MOBILE_ADVANCED_AUTH_MODE_PTR;
```

File

PrlEnums.h

PRL_MOUSE_BUTTON_PTR Type

Syntax

```
typedef PRL_MOUSE_BUTTON* PRL_MOUSE_BUTTON_PTR;
```

File

PrlEnums.h

PRL_NET_ADAPTER_EMULATED_TYPE_PTR Type

Syntax

```
typedef PRL_NET_ADAPTER_EMULATED_TYPE* PRL_NET_ADAPTER_EMULATED_TYPE_PTR;
```

File

PrlEnums.h

PRL_NET_ADAPTER_TYPE_PTR Type

Syntax

```
typedef PRL_NET_ADAPTER_TYPE* PRL_NET_ADAPTER_TYPE_PTR;
```

File

PrlEnums.h

PRL_NET_VIRTUAL_NETWORK_TYPE_PTR Type

Syntax

```
typedef PRL_NET_VIRTUAL_NETWORK_TYPE* PRL_NET_VIRTUAL_NETWORK_TYPE_PTR;
```

File

PrlEnums.h

PRL_NETFILTER_MODE_PTR Type

Syntax

```
typedef PRL_NETFILTER_MODE* PRL_NETFILTER_MODE_PTR;
```

File

PrlEnums.h

PRL_NETWORK_RATE_MPU_PTR Type

Syntax

```
typedef PRL_NETWORK_RATE_MPU* PRL_NETWORK_RATE_MPU_PTR;
```

File

PrlEnums.h

PRL_ONLINE_COMPACT_MODE_PTR Type

Syntax

```
typedef PRL_ONLINE_COMPACT_MODE* PRL_ONLINE_COMPACT_MODE_PTR;
```

File

PrlEnums.h

PRL_OPTIMIZE_MODIFIERS_MODE_PTR Type

Syntax

```
typedef PRL_OPTIMIZE_MODIFIERS_MODE* PRL_OPTIMIZE_MODIFIERS_MODE_PTR;
```

File

PrlEnums.h

PRL_PARAM_FIELD_DATA_TYPE_PTR Type

Syntax

```
typedef PRL_PARAM_FIELD_DATA_TYPE* PRL_PARAM_FIELD_DATA_TYPE_PTR;
```

File

PrlEnums.h

PRL_PORT_FORWARDING_TYPE_PTR Type

Syntax

```
typedef PRL_PORT_FORWARDING_TYPE* PRL_PORT_FORWARDING_TYPE_PTR;
```

File

PrlEnums.h

PRL_PRINTER_INTERFACE_TYPE_PTR Type

Syntax

```
typedef PRL_PRINTER_INTERFACE_TYPE* PRL_PRINTER_INTERFACE_TYPE_PTR;
```

File

PrlEnums.h

PRL_PRIVATE_EVENT_TYPE_PTR Type

Syntax

```
typedef PRL_PRIVATE_EVENT_TYPE* PRL_PRIVATE_EVENT_TYPE_PTR;
```

File

PrlEventsValues.h

PRL_PROBLEM_REPORT_FLAGS_PTR Type

Syntax

```
typedef PRL_PROBLEM_REPORT_FLAGS* PRL_PROBLEM_REPORT_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_PROBLEM_REPORT_REASON_PTR Type

Syntax

```
typedef PRL_PROBLEM_REPORT_REASON* PRL_PROBLEM_REPORT_REASON_PTR;
```

File

PrlEnums.h

PRL_PROBLEM_REPORT_SCHEME_PTR Type

Syntax

```
typedef PRL_PROBLEM_REPORT_SCHEME* PRL_PROBLEM_REPORT_SCHEME_PTR;
```

File

PrlEnums.h

PRL_PROBLEM_REPORT_TYPE_PTR Type

Syntax

```
typedef PRL_PROBLEM_REPORT_TYPE* PRL_PROBLEM_REPORT_TYPE_PTR;
```

File

PrlEnums.h

PRL_PROCESS_STATE_TYPE_PTR Type

Syntax

```
typedef PRL_PROCESS_STATE_TYPE* PRL_PROCESS_STATE_TYPE_PTR;
```

File

PrlEnums.h

PRL_PRODUCT_ENUM_PTR Type

Syntax

```
typedef PRL_PRODUCT_ENUM* PRL_PRODUCT_ENUM_PTR;
```

File

PrlEnums.h

PRL_PROXY_CONNECTION_STATUS_PTR Type

Syntax

```
typedef PRL_PROXY_CONNECTION_STATUS* PRL_PROXY_CONNECTION_STATUS_PTR;
```

File

PrlEnums.h

PRL_QUERY_ELEMENT_FLAGS_PTR Type

Syntax

```
typedef PRL_QUERY_ELEMENT_FLAGS* PRL_QUERY_ELEMENT_FLAGS_PTR;
```

File

PrlEnums.h

PRL_REGISTER_3PARTY_VM_FLAGS_PTR Type

Syntax

```
typedef PRL_REGISTER_3PARTY_VM_FLAGS* PRL_REGISTER_3PARTY_VM_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_REGISTER_NEW_VM_PTR Type

Syntax

```
typedef PRL_REGISTER_NEW_VM* PRL_REGISTER_NEW_VM_PTR;
```

File

PrlCommandsFlags.h

PRL_REGISTER_VM_FLAGS_PTR Type

Syntax

```
typedef PRL_REGISTER_VM_FLAGS* PRL_REGISTER_VM_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_RESIZE_IMAGE_FLAGS_PTR Type

Syntax

```
typedef PRL_RESIZE_IMAGE_FLAGS* PRL_RESIZE_IMAGE_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_RUNNING_TASK_TYPE_ENUM_PTR Type

Syntax

```
typedef PRL_RUNNING_TASK_TYPE_ENUM* PRL_RUNNING_TASK_TYPE_ENUM_PTR;
```

File

PrlEnums.h

PRL_SCALE_VIEW_MODE_PTR Type

Syntax

```
typedef PRL_SCALE_VIEW_MODE* PRL_SCALE_VIEW_MODE_PTR;
```

File

PrlEnums.h

PRL_SECURITY_LEVEL_PTR Type

Syntax

```
typedef PRL_SECURITY_LEVEL* PRL_SECURITY_LEVEL_PTR;
```

File

PrlCommandsFlags.h

PRL_SERIAL_PORT_SOCKET_OPERATION_MODE_PTR Type

Syntax

```
typedef PRL_SERIAL_PORT_SOCKET_OPERATION_MODE* PRL_SERIAL_PORT_SOCKET_OPERATION_MODE_PTR;
```

File

PrlEnums.h

PRL_SERVICE_STATUS_ENUM_PTR Type

Syntax

```
typedef PRL_SERVICE_STATUS_ENUM* PRL_SERVICE_STATUS_ENUM_PTR;
```

File

PrlEnums.h

PRL_SET_USERPASSWD_FLAGS_PTR Type

Syntax

```
typedef PRL_SET_USERPASSWD_FLAGS* PRL_SET_USERPASSWD_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_SIZE Type

Unsigned integer type of the result of the sizeof operator

Syntax

```
typedef unsigned int PRL_SIZE;
```

File

PrlTypes.h

PRL_SIZE_PTR Type

Syntax

```
typedef PRL_SIZE* PRL_SIZE_PTR;
```

File

PrlTypes.h

PRL_SIZE_PTR_PTR Type

Syntax

```
typedef PRL_SIZE_PTR* PRL_SIZE_PTR_PTR;
```

File

PrlTypes.h

PRL_SNAPSHOT_ELEMENT_PTR Type

Syntax

```
typedef PRL_SNAPSHOT_ELEMENT* PRL_SNAPSHOT_ELEMENT_PTR;
```

File

PrlDisk.h

PRL_SOUND_CHANNEL_TYPE_PTR Type

Syntax

```
typedef PRL_SOUND_CHANNEL_TYPE* PRL_SOUND_CHANNEL_TYPE_PTR;
```

File

PrlEnums.h

PRL_SRV_SHUTDOWN_FLAGS_PTR Type

Syntax

```
typedef PRL_SRV_SHUTDOWN_FLAGS* PRL_SRV_SHUTDOWN_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_STATES_CALLBACK_TYPE Type

Saved states callback types

@author antonz@

Syntax

```
typedef PRL_INT32 PRL_STATES_CALLBACK_TYPE;
```

File

PrlDisk.h

PRL_STATES_CALLBACK_TYPE_PTR Type

Syntax

```
typedef PRL_STATES_CALLBACK_TYPE* PRL_STATES_CALLBACK_TYPE_PTR;
```

File

PrlDisk.h

PRL_STR Type

Syntax

```
typedef PRL_CHAR_PTR PRL_STR;
```

File

PrlTypes.h

PRL_UNDO_DISKS_MODE_PTR Type

Syntax

```
typedef PRL_UNDO_DISKS_MODE* PRL_UNDO_DISKS_MODE_PTR;
```

File

PrlEnums.h

PRL_UPDATE_LICENSE_FLAGS_PTR Type

Syntax

```
typedef PRL_UPDATE_LICENSE_FLAGS* PRL_UPDATE_LICENSE_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_USB_DEVICE_AUTO_CONNECT_OPTION_PTR Type

Syntax

```
typedef PRL_USB_DEVICE_AUTO_CONNECT_OPTION* PRL_USB_DEVICE_AUTO_CONNECT_OPTION_PTR;
```

File

PrlEnums.h

PRL_USB_DEVICE_CONNECT_REASON_PTR Type

Syntax

```
typedef PRL_USB_DEVICE_CONNECT_REASON* PRL_USB_DEVICE_CONNECT_REASON_PTR;
```

File

PrlEnums.h

PRL_USB_DEVICE_TYPE_PTR Type

Syntax

```
typedef PRL_USB_DEVICE_TYPE* PRL_USB_DEVICE_TYPE_PTR;
```

File

PrlEnums.h

PRL_VCMMD_COMMAND_FLAGS_PTR Type

Syntax

```
typedef PRL_VCMMD_COMMAND_FLAGS* PRL_VCMMD_COMMAND_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_VERSION_ENUM_PTR Type

Syntax

```
typedef PRL_VERSION_ENUM* PRL_VERSION_ENUM_PTR;
```

File

PrlEnums.h

PRL_VIDEO_3D_ACCELERATION_PTR Type

Syntax

```
typedef PRL_VIDEO_3D_ACCELERATION* PRL_VIDEO_3D_ACCELERATION_PTR;
```

File

PrlEnums.h

PRL_VIDEO_RECEIVER_CONNECTION_STATE_PTR Type

Syntax

```
typedef PRL_VIDEO_RECEIVER_CONNECTION_STATE* PRL_VIDEO_RECEIVER_CONNECTION_STATE_PTR;
```

File

PrlEnums.h

PRL_VIRTUAL_MACHINE_PROFILE_PTR Type

Syntax

```
typedef PRL_VIRTUAL_MACHINE_PROFILE* PRL_VIRTUAL_MACHINE_PROFILE_PTR;
```

File

PrlEnums.h

PRL_VIRTUAL_NETWORK_FLAGS_PTR Type

Syntax

```
typedef PRL_VIRTUAL_NETWORK_FLAGS* PRL_VIRTUAL_NETWORK_FLAGS_PTR;
```

File

PrlCommandsFlags.h

PRL_VMCTG_EVENT_PTR Type

Syntax

```
typedef PRL_VMCTG_EVENT* PRL_VMCTG_EVENT_PTR;
```

File

PrlEnums.h

PRL_VTD_STATUS_CODE_PTR Type

Syntax

```
typedef PRL_VTD_STATUS_CODE* PRL_VTD_STATUS_CODE_PTR;
```

File

PrlEnums.h

PRL_WINDOW_CORNERS_PTR Type

Syntax

```
typedef PRL_WINDOW_CORNERS* PRL_WINDOW_CORNERS_PTR;
```

File

PrlEnums.h

PRL_WINDOW_EDGE_MASK_PTR Type

Syntax

```
typedef PRL_WINDOW_EDGE_MASK * PRL_WINDOW_EDGE_MASK_PTR;
```

File

PrlEnums.h

Other Types

Types

Name	Description
IPluginInfoPtr (p. 835)	
VIRTUAL_MACHINE_ADDITION_STATE_PTR (p. 836)	
VIRTUAL_MACHINE_STATE_PTR (p. 836)	

IPluginInfoPtr Type

Syntax

```
typedef IPluginInfo* IPluginInfoPtr;
```

File

PrlPluginClasses.h

VIRTUAL_MACHINE_ADDITION_STATE_PTR Type

Syntax

```
typedef VIRTUAL_MACHINE_ADDITION_STATE* VIRTUAL_MACHINE_ADDITION_STATE_PTR;
```

File

PrlEnums.h

VIRTUAL_MACHINE_STATE_PTR Type

Syntax

```
typedef VIRTUAL_MACHINE_STATE* VIRTUAL_MACHINE_STATE_PTR;
```

File

PrlEnums.h

Structs, Records, Enums

Topics

Name	Description
PRL_IO Structs (p. 837)	
PRL_VM Structs (p. 920)	
Other PRL Structs (p. 953)	
Other Structs (p. 1102)	

PRL_IO Structs

Enumerations

	Name	Description
	_PRL_IO_CLIENT_FLAGS (p. 837)	Flags which determines peer client while login to dispatcher and while attach to vm.
	_PRL_IO_DISCONNECT_REASON (p. 838)	Specifies the reason of closing IO connection
	_PRL_IO_STATE (p. 917)	
	PRL_IO_CLIENT_FLAGS (p. 918)	Flags which determines peer client while login to dispatcher and while attach to vm.
	PRL_IO_DISCONNECT_REASON (p. 918)	Specifies the reason of closing IO connection
	PRL_IO_STATE (p. 919)	
	PRL_IOLIMIT_TYPE (p. 920)	

Structures

	Name	Description
	_PRL_IO_DISPLAY_SCREEN_SIZE (p. 838)	
	_PRL_IO_DPI_DISPLAY_SCREEN_SIZE (p. 839)	
	_PRL_IO_SCREEN_SIZE (p. 917)	
	PRL_IO_DISPLAY_SCREEN_SIZE (p. 919)	
	PRL_IO_SCREEN_SIZE (p. 919)	

_PRL_IO_CLIENT_FLAGS Enumeration

Flags which determines peer client while login to dispatcher and while attach to vm.

Syntax

```
enum _PRL_IO_CLIENT_FLAGS {
  PCF_ORIGINAL_CLIENT = 1<<(PACF_MAX+1),
  PCF_LIGHTWEIGHT_CLIENT = 1<<(PACF_MAX+2),
  PCF_TRUSTED_CHANNEL = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

Members

PCF_ORIGINAL_CLIENT

Client with all features

PCF_LIGHTWEIGHT_CLIENT

- Lightweight client (iPhone/iPad) which

does not subscribe on heavy packages

PCF_TRUSTED_CHANNEL

- Login over trusted channel

(no need in username/password)

_PRL_IO_DISCONNECT_REASON Enumeration

Specifies the reason of closing IO connection

Syntax

```
enum _PRL_IO_DISCONNECT_REASON {
  PDR_UNSPECIFIED = 0,
  PDR_RECONNECT = 1,
  PDR_HOST_USER_LOGGED_IN = 2,
  PDR_HOST_USER_INPUT_DETECTED = 3,
  PDR_ANOTHER_CLIENT_CONNECTED = 4,
  PDR_SESSION_BECOME_INACTIVE = 5
};
```

File

PrlEnums.h

_PRL_IO_DISPLAY_SCREEN_SIZE Structure

Syntax

```
struct _PRL_IO_DISPLAY_SCREEN_SIZE {
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 depth;
  PRL_INT32 lineBytes;
  PRL_UINT32 display;
  PRL_INT32 displayX;
  PRL_INT32 displayY;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Structure

Syntax

```
struct _PRL_IO_DPI_DISPLAY_SCREEN_SIZE {
  PRL_IO_DISPLAY_SCREEN_SIZE display;
  PRL_INT32 dpi;
  struct _PRL_IO_DISPLAY_PALETTE {
    PRL_IO_PALETTE palette;
    PRL_UINT32 display;
  };
  struct _PRL_IO_DISPLAY_GAMMA {
    PRL_UINT32 display;
    PRL_UINT32 type;
    PRL_UINT32 gammaSize;
    PRL_UINT32 reserved;
  };
  struct _PRL_IO_SCREEN_REGION {
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
  };
  struct _PRL_IO_DISPLAY_SCREEN_REGION {
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
  };
  struct _PRL_IO_SCREEN_BUFFER_REGION {
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 scaleWidth;
    PRL_UINT32 scaleHeight;
    PRL_UINT32 display;
  };
  struct _PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION {
    PRL_BOOL success;
    struct _PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX {
      PRL_BOOL success;
      struct _PRL_IO_KEYBOARD_SCANCODE {
        PRL_UINT16 key;
      };
      struct _PRL_IO_KEYBOARD_LEDS {
        PRL_UINT32 ledsState;
      };
      struct _PRL_IO_MOUSE_EVENT {
        PRL_INT32 x;
        PRL_INT32 y;
        PRL_INT32 z;
        PRL_UINT32 buttons;
      };
      struct _PRL_IO_MOUSE_EVENT_EX {
        PRL_INT32 x;
        PRL_INT32 y;
```

```
  PRL_INT32 z;
  PRL_UINT32 buttons;
  PRL_INT32 w;
};
struct _PRL_IO_MOUSE_EVENT_EX2 {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
  PRL_UINT32 buttons;
  PRL_INT32 w;
  PRL_INT32 z120;
  PRL_INT32 w120;
};
struct _PRL_IO_GRACEFUL_SHUTDOWN {
  PRL_UINT32 kind;
};
struct _PRL_IO_DESKTOP_UTILITIES_STATE {
  PRL_UINT32 bDesktopShown;
  PRL_UINT32 bTaskbarShown;
};
struct _PRL_IO_LANGUAGE_HOTKEYS {
  PRL_UINT32 LanguageHotkey;
  PRL_UINT32 LayoutHotkey;
};
struct _PRL_IO_SCREEN_SURFACE {
  PRL_UINT32 windowId;
  PRL_UINT32 surfaceId;
  PRL_INT32 scrollX;
  PRL_INT32 scrollY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
};
struct _PRL_IO_MOUSE_CURSOR {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 hotSpotX;
  PRL_UINT32 hotSpotY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  struct _PRL_IO_SLIDING_MOUSE {
    PRL_UINT32 state;
  };
  struct _PRL_IO_AVAILABLE_DISPLAYS {
    PRL_UINT32 availableDisplays;
  };
  struct _PRL_IO_DEVICE_USED {
    PRL_UINT32 devType;
    PRL_UINT32 devIndex;
    PRL_UINT32 devAction;
  };
  struct _PRL_IO_GENERIC_BUFFER {
    PRL_UINT32 size;
    PRL_VOID_PTR data;
  };
  struct _PRL_TIS_VERSION {
    struct _PRL_TIS_INT_VERSION {
      struct _PRL_TIS_RECORD_INFO {
        struct _PRL_IO_TOOLS_UTILITY_COMMAND {
          struct _PRL_IO_TOOLS_SIA_DATA {
            struct _PRL_IO_DYNRES_TOOLS_STATUS {
              PRL_UINT32 status;
            };
            struct _PRL_PROXY_CLIENT_INFO {
              PRL_UINT32 structSize;
              PRL_CHAR clientOptions[64];
```

```
  PRL_UINT8 customerSupportUUID[16];
  PRL_CHAR managerURL[256];
};
struct _PRL_PROXY_REGISTRATION_INFO {
  PRL_CHAR hostAddresses[16][64];
  PRL_INT32 hostAddressesSz;
  PRL_CHAR hostName[64];
  PRL_UINT8 appUUID[16];
  PRL_CHAR appVersion[32];
  PRL_CHAR hostOptions[33];
  PRL_CHAR hostOsName[63];
  PRL_CHAR isoTimestamp[32];
};
struct _PRL_PROXY_HOST_INFO {
  PRL_UINT8 hostUUID[16];
  PRL_CHAR brokerAddresses[16][64];
  PRL_INT32 brokerAddressesSz;
  PRL_PROXY_REGISTRATION_INFO info;
};
struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
```

```
                    PRL_UINT32 attachResult;
                    PRL_UINT32 encodingResult;
                  };
                  struct _PRL_IO_SCREEN_UPDATE {
                    PRL_UINT32 offset;
                    PRL_UINT32 x;
                    PRL_UINT32 y;
                    PRL_UINT32 width;
                    PRL_UINT32 height;
                    PRL_UINT32 display;
                    struct _PRL_STAT_NET_TRAFFIC {
                      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
                      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
                      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
                      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
                    };
                    struct _PRL_CPULIMIT_DATA {
                      PRL_UINT32 value;
                      PRL_CPULIMIT_TYPE type;
                    };
                    struct _PRL_IOLIMIT_DATA {
                      PRL_UINT32 value;
                      PRL_IOLIMIT_TYPE type;
                    };
                    struct _PRL_GET_VM_CONFIG_PARAM_DATA {
                      PRL_VM_TYPE nVmType;
                      PRL_CONST_STR sConfigSample;
                      PRL_UINT32 nOsVersion;
                    };
                    struct _PRL_CVSRC_OPEN_MODE {
                      PRL_UINT32 sessionId;
                      PRL_UINT32 frameFormat;
                      PRL_UINT32 frameWidth;
                      PRL_UINT32 frameHeight;
                      PRL_UINT32 frameBytesPerRow;
                    };
                    struct _PRL_IO_SSO_CLIENT_VM_PACKET {
                      PRL_UINT8 credentialsRequestUUID[16];
                      PRL_UINT32 reserved[12];
                    };
                    struct _PRL_IO_CHANNEL_STAT {
                      struct _PRL_CPU_FEATURES_MASK {
                        struct _PRL_CPU_FEATURES {
                          struct _PRL_HOST_OS_INFO {
                            struct _PRL_MEMGUARANTEE_DATA {
                              PRL_UINT32 value;
                              PRL_MEMGUARANTEE_TYPE type;
                            };
                          };
                        };
                      };
                    };
                  };
                };
              };
            };
          };
        };
      };
    };
  };
```

File

PrlIOStructs.h

Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Data Members

	Name	Description
	display (p. 844)	
	dpi (p. 844)	

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Nested Types

Name	Description
PRL_IO_DISPLAY_GAMMA (p. 844)	
PRL_IO_DISPLAY_GAMMA_PTR (p. 845)	
PRL_IO_DISPLAY_PALETTE (p. 845)	
PRL_IO_DISPLAY_PALETTE_PTR (p. 845)	
PRL_IO_DISPLAY_SCREEN_REGION (p. 845)	
PRL_IO_DISPLAY_SCREEN_REGION_PTR (p. 845)	
PRL_IO_DPI_DISPLAY_SCREEN_SIZE_PTR (p. 845)	
PRL_IO_SCREEN_BUFFER_REGION (p. 846)	
PRL_IO_SCREEN_BUFFER_REGION_PTR (p. 846)	
PRL_IO_SCREEN_REGION (p. 846)	
PRL_IO_SCREEN_REGION_PTR (p. 846)	

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Structures

	Name	Description
	_PRL_IO_DISPLAY_GAMMA (p. 846)	
	_PRL_IO_DISPLAY_PALETTE (p. 847)	
	_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION (p. 847)	
	_PRL_IO_DISPLAY_SCREEN_REGION (p. 916)	
	_PRL_IO_SCREEN_BUFFER_REGION (p. 916)	
	_PRL_IO_SCREEN_REGION (p. 917)	

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Data Members

	Name	Description
	display (p. 844)	
	dpi (p. 844)	

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Nested Types

Name	Description
PRL_IO_DISPLAY_GAMMA (p. 844)	
PRL_IO_DISPLAY_GAMMA_PTR (p. 845)	
PRL_IO_DISPLAY_PALETTE (p. 845)	
PRL_IO_DISPLAY_PALETTE_PTR (p. 845)	
PRL_IO_DISPLAY_SCREEN_REGION (p. 845)	

PRL_IO_DISPLAY_SCREEN_REGION_PTR (p. 845)	
PRL_IO_DPI_DISPLAY_SCREEN_SIZE_PTR (p. 845)	
PRL_IO_SCREEN_BUFFER_REGION (p. 846)	
PRL_IO_SCREEN_BUFFER_REGION_PTR (p. 846)	
PRL_IO_SCREEN_REGION (p. 846)	
PRL_IO_SCREEN_REGION_PTR (p. 846)	

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Structures

	Name	Description
	_PRL_IO_DISPLAY_GAMMA (p. 846)	
	_PRL_IO_DISPLAY_PALETTE (p. 847)	
	_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION (p. 847)	
	_PRL_IO_DISPLAY_SCREEN_REGION (p. 916)	
	_PRL_IO_SCREEN_BUFFER_REGION (p. 916)	
	_PRL_IO_SCREEN_REGION (p. 917)	

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Data Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::display Data Member

Syntax

```
PRL_IO_DISPLAY_SCREEN_SIZE display;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::dpi Data Member

Syntax

```
PRL_INT32 dpi;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_GAMMA Nested Type

Syntax

```
typedef struct _PRL_IO_DISPLAY_GAMMA {
  PRL_UINT32 display;
  PRL_UINT32 type;
  PRL_UINT32 gammaSize;
  PRL_UINT32 reserved;
} PRL_IO_DISPLAY_GAMMA;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_GAMMA_PTR Nested Type

Syntax

```
typedef PRL_IO_DISPLAY_GAMMA* PRL_IO_DISPLAY_GAMMA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_PALETTE Nested Type

Syntax

```
typedef struct _PRL_IO_DISPLAY_PALETTE {
  PRL_IO_PALETTE palette;
  PRL_UINT32 display;
} PRL_IO_DISPLAY_PALETTE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_PALETTE_PTR Nested Type

Syntax

```
typedef PRL_IO_DISPLAY_PALETTE* PRL_IO_DISPLAY_PALETTE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_SCREEN_REGION Nested Type

Syntax

```
typedef struct _PRL_IO_DISPLAY_SCREEN_REGION {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
} PRL_IO_DISPLAY_SCREEN_REGION;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_SCREEN_REGION_PTR Nested Type

Syntax

```
typedef PRL_IO_DISPLAY_SCREEN_REGION* PRL_IO_DISPLAY_SCREEN_REGION_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DPI_DISPLAY_SCREEN_SIZE_PTR Nested Type

Syntax

```
typedef PRL_IO_DPI_DISPLAY_SCREEN_SIZE* PRL_IO_DPI_DISPLAY_SCREEN_SIZE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_BUFFER_REGION Nested Type

Syntax

```
typedef struct _PRL_IO_SCREEN_BUFFER_REGION {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 scaleWidth;
  PRL_UINT32 scaleHeight;
  PRL_UINT32 display;
} PRL_IO_SCREEN_BUFFER_REGION;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_BUFFER_REGION_PTR Nested Type

Syntax

```
typedef PRL_IO_SCREEN_BUFFER_REGION* PRL_IO_SCREEN_BUFFER_REGION_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_REGION Nested Type

Syntax

```
typedef struct _PRL_IO_SCREEN_REGION {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
} PRL_IO_SCREEN_REGION;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_REGION_PTR Nested Type

Syntax

```
typedef PRL_IO_SCREEN_REGION* PRL_IO_SCREEN_REGION_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_GAMMA Structure

Syntax

```
struct _PRL_IO_DISPLAY_GAMMA {
  PRL_UINT32 display;
  PRL_UINT32 type;
  PRL_UINT32 gammaSize;
  PRL_UINT32 reserved;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_PALETTE Structure

Syntax

```
struct _PRL_IO_DISPLAY_PALETTE {
  PRL_IO_PALETTE palette;
  PRL_UINT32 display;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Structure

Syntax

```
struct _PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION {
  PRL_BOOL success;
  struct _PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX {
    PRL_BOOL success;
    struct _PRL_IO_KEYBOARD_SCANCODE {
      PRL_UINT16 key;
    };
    struct _PRL_IO_KEYBOARD_LEDS {
      PRL_UINT32 ledsState;
    };
    struct _PRL_IO_MOUSE_EVENT {
      PRL_INT32 x;
      PRL_INT32 y;
      PRL_INT32 z;
      PRL_UINT32 buttons;
    };
    struct _PRL_IO_MOUSE_EVENT_EX {
      PRL_INT32 x;
      PRL_INT32 y;
      PRL_INT32 z;
      PRL_UINT32 buttons;
      PRL_INT32 w;
    };
    struct _PRL_IO_MOUSE_EVENT_EX2 {
      PRL_INT32 x;
      PRL_INT32 y;
      PRL_INT32 z;
      PRL_UINT32 buttons;
      PRL_INT32 w;
      PRL_INT32 z120;
      PRL_INT32 w120;
    };
    struct _PRL_IO_GRACEFUL_SHUTDOWN {
      PRL_UINT32 kind;
    };
    struct _PRL_IO_DESKTOP_UTILITIES_STATE {
      PRL_UINT32 bDesktopShown;
```

```
  PRL_UINT32 bTaskbarShown;
};
struct _PRL_IO_LANGUAGE_HOTKEYS {
  PRL_UINT32 LanguageHotkey;
  PRL_UINT32 LayoutHotkey;
};
struct _PRL_IO_SCREEN_SURFACE {
  PRL_UINT32 windowId;
  PRL_UINT32 surfaceId;
  PRL_INT32 scrollX;
  PRL_INT32 scrollY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
};
struct _PRL_IO_MOUSE_CURSOR {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 hotSpotX;
  PRL_UINT32 hotSpotY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  struct _PRL_IO_SLIDING_MOUSE {
    PRL_UINT32 state;
  };
  struct _PRL_IO_AVAILABLE_DISPLAYS {
    PRL_UINT32 availableDisplays;
  };
  struct _PRL_IO_DEVICE_USED {
    PRL_UINT32 devType;
    PRL_UINT32 devIndex;
    PRL_UINT32 devAction;
  };
  struct _PRL_IO_GENERIC_BUFFER {
    PRL_UINT32 size;
    PRL_VOID_PTR data;
  };
  struct _PRL_TIS_VERSION {
    struct _PRL_TIS_INT_VERSION {
      struct _PRL_TIS_RECORD_INFO {
        struct _PRL_IO_TOOLS_UTILITY_COMMAND {
          struct _PRL_IO_TOOLS_SIA_DATA {
            struct _PRL_IO_DYNRES_TOOLS_STATUS {
              PRL_UINT32 status;
            };
            struct _PRL_PROXY_CLIENT_INFO {
              PRL_UINT32 structSize;
              PRL_CHAR clientOptions[64];
              PRL_UINT8 customerSupportUUID[16];
              PRL_CHAR managerURL[256];
            };
            struct _PRL_PROXY_REGISTRATION_INFO {
              PRL_CHAR hostAddresses[16][64];
              PRL_INT32 hostAddressesSz;
              PRL_CHAR hostName[64];
              PRL_UINT8 appUUID[16];
              PRL_CHAR appVersion[32];
              PRL_CHAR hostOptions[33];
              PRL_CHAR hostOsName[63];
              PRL_CHAR isoTimestamp[32];
            };
            struct _PRL_PROXY_HOST_INFO {
              PRL_UINT8 hostUUID[16];
              PRL_CHAR brokerAddresses[16][64];
              PRL_INT32 brokerAddressesSz;
              PRL_PROXY_REGISTRATION_INFO info;
```

```
};
struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
    };
    struct _PRL_CPULIMIT_DATA {
      PRL_UINT32 value;
```

```
    PRL_CPULIMIT_TYPE type;
};
struct _PRL_IOLIMIT_DATA {
    PRL_UINT32 value;
    PRL_IOLIMIT_TYPE type;
};
struct _PRL_GET_VM_CONFIG_PARAM_DATA {
    PRL_VM_TYPE nVmType;
    PRL_CONST_STR sConfigSample;
    PRL_UINT32 nOsVersion;
};
struct _PRL_CVSRC_OPEN_MODE {
    PRL_UINT32 sessionId;
    PRL_UINT32 frameFormat;
    PRL_UINT32 frameWidth;
    PRL_UINT32 frameHeight;
    PRL_UINT32 frameBytesPerRow;
};
struct _PRL_IO_SSO_CLIENT_VM_PACKET {
    PRL_UINT8 credentialsRequestUUID[16];
    PRL_UINT32 reserved[12];
};
struct _PRL_IO_CHANNEL_STAT {
    struct _PRL_CPU_FEATURES_MASK {
        struct _PRL_CPU_FEATURES {
            struct _PRL_HOST_OS_INFO {
                struct _PRL_MEMGUARANTEE_DATA {
                    PRL_UINT32 value;
                    PRL_MEMGUARANTEE_TYPE type;
                };
            };
        };
    };
};
```

File

PrlIOStructs.h

Members

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Data Members

	Name	Description
	success (p. 851)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Structures

	Name	Description
	_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX (p. 851)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Data Members

	Name	Description
	success (p. 851)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Structures

	Name	Description
	_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX (p. 851)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Data Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::success Data Member

Syntax

```
PRL_BOOL success;
```

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Structure

Syntax

```
struct _PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX {
  PRL_BOOL success;
  struct _PRL_IO_KEYBOARD_SCANCODE {
    PRL_UINT16 key;
  };
  struct _PRL_IO_KEYBOARD_LEDS {
    PRL_UINT32 ledsState;
  };
  struct _PRL_IO_MOUSE_EVENT {
    PRL_INT32 x;
    PRL_INT32 y;
    PRL_INT32 z;
    PRL_UINT32 buttons;
  };
  struct _PRL_IO_MOUSE_EVENT_EX {
    PRL_INT32 x;
    PRL_INT32 y;
    PRL_INT32 z;
    PRL_UINT32 buttons;
    PRL_INT32 w;
  };
  struct _PRL_IO_MOUSE_EVENT_EX2 {
    PRL_INT32 x;
    PRL_INT32 y;
    PRL_INT32 z;
    PRL_UINT32 buttons;
    PRL_INT32 w;
    PRL_INT32 z120;
    PRL_INT32 w120;
  };
  struct _PRL_IO_GRACEFUL_SHUTDOWN {
    PRL_UINT32 kind;
  };
  struct _PRL_IO_DESKTOP_UTILITIES_STATE {
    PRL_UINT32 bDesktopShown;
    PRL_UINT32 bTaskbarShown;
  };
  struct _PRL_IO_LANGUAGE_HOTKEYS {
    PRL_UINT32 LanguageHotkey;
```

```
  PRL_UINT32 LayoutHotkey;
};
struct _PRL_IO_SCREEN_SURFACE {
  PRL_UINT32 windowId;
  PRL_UINT32 surfaceId;
  PRL_INT32 scrollX;
  PRL_INT32 scrollY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
};
struct _PRL_IO_MOUSE_CURSOR {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 hotSpotX;
  PRL_UINT32 hotSpotY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  struct _PRL_IO_SLIDING_MOUSE {
    PRL_UINT32 state;
  };
  struct _PRL_IO_AVAILABLE_DISPLAYS {
    PRL_UINT32 availableDisplays;
  };
  struct _PRL_IO_DEVICE_USED {
    PRL_UINT32 devType;
    PRL_UINT32 devIndex;
    PRL_UINT32 devAction;
  };
  struct _PRL_IO_GENERIC_BUFFER {
    PRL_UINT32 size;
    PRL_VOID_PTR data;
  };
  struct _PRL_TIS_VERSION {
    struct _PRL_TIS_INT_VERSION {
      struct _PRL_TIS_RECORD_INFO {
        struct _PRL_IO_TOOLS_UTILITY_COMMAND {
          struct _PRL_IO_TOOLS_SIA_DATA {
            struct _PRL_IO_DYNRES_TOOLS_STATUS {
              PRL_UINT32 status;
            };
            struct _PRL_PROXY_CLIENT_INFO {
              PRL_UINT32 structSize;
              PRL_CHAR clientOptions[64];
              PRL_UINT8 customerSupportUUID[16];
              PRL_CHAR managerURL[256];
            };
            struct _PRL_PROXY_REGISTRATION_INFO {
              PRL_CHAR hostAddresses[16][64];
              PRL_INT32 hostAddressesSz;
              PRL_CHAR hostName[64];
              PRL_UINT8 appUUID[16];
              PRL_CHAR appVersion[32];
              PRL_CHAR hostOptions[33];
              PRL_CHAR hostOsName[63];
              PRL_CHAR isoTimestamp[32];
            };
            struct _PRL_PROXY_HOST_INFO {
              PRL_UINT8 hostUUID[16];
              PRL_CHAR brokerAddresses[16][64];
              PRL_INT32 brokerAddressesSz;
              PRL_PROXY_REGISTRATION_INFO info;
            };
            struct _PRL_PROXY_GEO_INFO {
              PRL_CHAR countryName[64];
              PRL_CHAR countryCode[3];
```

```
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
    };
    struct _PRL_CPULIMIT_DATA {
      PRL_UINT32 value;
      PRL_CPULIMIT_TYPE type;
    };
    struct _PRL_IOLIMIT_DATA {
      PRL_UINT32 value;
```

```
                PRL_IOLIMIT_TYPE type;
              };
              struct _PRL_GET_VM_CONFIG_PARAM_DATA {
                PRL_VM_TYPE nVmType;
                PRL_CONST_STR sConfigSample;
                PRL_UINT32 nOsVersion;
              };
              struct _PRL_CVSRC_OPEN_MODE {
                PRL_UINT32 sessionId;
                PRL_UINT32 frameFormat;
                PRL_UINT32 frameWidth;
                PRL_UINT32 frameHeight;
                PRL_UINT32 frameBytesPerRow;
              };
              struct _PRL_IO_SSO_CLIENT_VM_PACKET {
                PRL_UINT8 credentialsRequestUUID[16];
                PRL_UINT32 reserved[12];
              };
              struct _PRL_IO_CHANNEL_STAT {
                struct _PRL_CPU_FEATURES_MASK {
                  struct _PRL_CPU_FEATURES {
                    struct _PRL_HOST_OS_INFO {
                      struct _PRL_MEMGUARANTEE_DATA {
                        PRL_UINT32 value;
                        PRL_MEMGUARANTEE_TYPE type;
                      };
                    };
                  };
                };
              };
            };
          };
        };
      };
    };
  };
};
```

File

PrlIOStructs.h

Members

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Data Members

	Name	Description
	success (p. 856)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Nested Types

Name	Description
PRL_IO_DESKTOP_UTILITIES_STATE (p. 856)	
PRL_IO_DESKTOP_UTILITIES_STATE_PTR (p. 857)	
PRL_IO_GRACEFUL_SHUTDOWN (p. 857)	
PRL_IO_GRACEFUL_SHUTDOWN_PTR (p. 857)	
PRL_IO_KEYBOARD_LEDS (p. 857)	
PRL_IO_KEYBOARD_LEDS_PTR (p. 857)	
PRL_IO_KEYBOARD_SCANCODE (p. 857)	

PRL_IO_KEYBOARD_SCANCODE_PTR (p. 857)	
PRL_IO_LANGUAGE_HOTKEYS (p. 858)	
PRL_IO_LANGUAGE_HOTKEYS_PTR (p. 858)	
PRL_IO_MOUSE_EVENT (p. 858)	
PRL_IO_MOUSE_EVENT_EX (p. 858)	
PRL_IO_MOUSE_EVENT_EX_PTR (p. 858)	
PRL_IO_MOUSE_EVENT_EX2 (p. 858)	
PRL_IO_MOUSE_EVENT_EX2_PTR (p. 859)	
PRL_IO_MOUSE_EVENT_PTR (p. 859)	
PRL_IO_SCREEN_SURFACE (p. 859)	
PRL_IO_SCREEN_SURFACE_PTR (p. 859)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Structures

	Name	Description
	_PRL_IO_DESKTOP_UTILITIES_STATE (p. 859)	
	_PRL_IO_GRACEFUL_SHUTDOWN (p. 860)	
	_PRL_IO_KEYBOARD_LEDS (p. 860)	
	_PRL_IO_KEYBOARD_SCANCODE (p. 860)	
	_PRL_IO_LANGUAGE_HOTKEYS (p. 860)	
	_PRL_IO_MOUSE_CURSOR (p. 860)	
	_PRL_IO_MOUSE_EVENT (p. 915)	
	_PRL_IO_MOUSE_EVENT_EX (p. 915)	
	_PRL_IO_MOUSE_EVENT_EX2 (p. 915)	
	_PRL_IO_SCREEN_SURFACE (p. 916)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Data Members

	Name	Description
	success (p. 856)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Nested Types

Name	Description
PRL_IO_DESKTOP_UTILITIES_STATE (p. 856)	
PRL_IO_DESKTOP_UTILITIES_STATE_PTR (p. 857)	
PRL_IO_GRACEFUL_SHUTDOWN (p. 857)	
PRL_IO_GRACEFUL_SHUTDOWN_PTR (p. 857)	
PRL_IO_KEYBOARD_LEDS (p. 857)	
PRL_IO_KEYBOARD_LEDS_PTR (p. 857)	
PRL_IO_KEYBOARD_SCANCODE (p. 857)	
PRL_IO_KEYBOARD_SCANCODE_PTR (p. 857)	
PRL_IO_LANGUAGE_HOTKEYS (p. 858)	
PRL_IO_LANGUAGE_HOTKEYS_PTR (p. 858)	

PRL_IO_MOUSE_EVENT (p. 858)	
PRL_IO_MOUSE_EVENT_EX (p. 858)	
PRL_IO_MOUSE_EVENT_EX_PTR (p. 858)	
PRL_IO_MOUSE_EVENT_EX2 (p. 858)	
PRL_IO_MOUSE_EVENT_EX2_PTR (p. 859)	
PRL_IO_MOUSE_EVENT_PTR (p. 859)	
PRL_IO_SCREEN_SURFACE (p. 859)	
PRL_IO_SCREEN_SURFACE_PTR (p. 859)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Structures

	Name	Description
	_PRL_IO_DESKTOP_UTILITIES_STATE (p. 859)	
	_PRL_IO_GRACEFUL_SHUTDOWN (p. 860)	
	_PRL_IO_KEYBOARD_LEDS (p. 860)	
	_PRL_IO_KEYBOARD_SCANCODE (p. 860)	
	_PRL_IO_LANGUAGE_HOTKEYS (p. 860)	
	_PRL_IO_MOUSE_CURSOR (p. 860)	
	_PRL_IO_MOUSE_EVENT (p. 915)	
	_PRL_IO_MOUSE_EVENT_EX (p. 915)	
	_PRL_IO_MOUSE_EVENT_EX2 (p. 915)	
	_PRL_IO_SCREEN_SURFACE (p. 916)	

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Data Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::success
Data Member

Syntax

```
PRL_BOOL success;
```

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_DESKTOP_UTILITIES_STATE
Nested Type

Syntax

```
typedef struct _PRL_IO_DESKTOP_UTILITIES_STATE {
  PRL_UINT32 bDesktopShown;
  PRL_UINT32 bTaskbarShown;
} PRL_IO_DESKTOP_UTILITIES_STATE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_DESKTOP_UTILITIES_STATE_PTR
Nested Type

Syntax

```
typedef PRL_IO_DESKTOP_UTILITIES_STATE* PRL_IO_DESKTOP_UTILITIES_STATE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_GRACEFUL_SHUTDOWN
Nested Type

Syntax

```
typedef struct _PRL_IO_GRACEFUL_SHUTDOWN {
  PRL_UINT32 kind;
} PRL_IO_GRACEFUL_SHUTDOWN;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_GRACEFUL_SHUTDOWN_PTR
Nested Type

Syntax

```
typedef PRL_IO_GRACEFUL_SHUTDOWN* PRL_IO_GRACEFUL_SHUTDOWN_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_LEDS
Nested Type

Syntax

```
typedef struct _PRL_IO_KEYBOARD_LEDS {
  PRL_UINT32 ledsState;
} PRL_IO_KEYBOARD_LEDS;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_LEDS_PTR
Nested Type

Syntax

```
typedef PRL_IO_KEYBOARD_LEDS* PRL_IO_KEYBOARD_LEDS_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_SCANCODE
Nested Type

Syntax

```
typedef struct _PRL_IO_KEYBOARD_SCANCODE {
  PRL_UINT16 key;
} PRL_IO_KEYBOARD_SCANCODE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_SCANCODE_PTR
Nested Type

Syntax

```
typedef PRL_IO_KEYBOARD_SCANCODE* PRL_IO_KEYBOARD_SCANCODE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_LANGUAGE_HOTKEYS
Nested Type

Syntax

```
typedef struct _PRL_IO_LANGUAGE_HOTKEYS {
  PRL_UINT32 LanguageHotkey;
  PRL_UINT32 LayoutHotkey;
} PRL_IO_LANGUAGE_HOTKEYS;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_LANGUAGE_HOTKEYS_PTR
Nested Type

Syntax

```
typedef PRL_IO_LANGUAGE_HOTKEYS* PRL_IO_LANGUAGE_HOTKEYS_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT
Nested Type

Syntax

```
typedef struct _PRL_IO_MOUSE_EVENT {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
  PRL_UINT32 buttons;
} PRL_IO_MOUSE_EVENT;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX
Nested Type

Syntax

```
typedef struct _PRL_IO_MOUSE_EVENT_EX {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
  PRL_UINT32 buttons;
  PRL_INT32 w;
} PRL_IO_MOUSE_EVENT_EX;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX_PTR
Nested Type

Syntax

```
typedef PRL_IO_MOUSE_EVENT_EX* PRL_IO_MOUSE_EVENT_EX_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX2
Nested Type

Syntax

```
typedef struct _PRL_IO_MOUSE_EVENT_EX2 {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
```

```
    PRL_UINT32 buttons;
    PRL_INT32 w;
    PRL_INT32 z120;
    PRL_INT32 w120;
  } PRL_IO_MOUSE_EVENT_EX2;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX2_PTR
Nested Type

Syntax

```
typedef PRL_IO_MOUSE_EVENT_EX2* PRL_IO_MOUSE_EVENT_EX2_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_PTR
Nested Type

Syntax

```
typedef PRL_IO_MOUSE_EVENT* PRL_IO_MOUSE_EVENT_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_SCREEN_SURFACE
Nested Type

Syntax

```
typedef struct _PRL_IO_SCREEN_SURFACE {
  PRL_UINT32 windowId;
  PRL_UINT32 surfaceId;
  PRL_INT32 scrollX;
  PRL_INT32 scrollY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
} PRL_IO_SCREEN_SURFACE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_SCREEN_SURFACE_PTR
Nested Type

Syntax

```
typedef PRL_IO_SCREEN_SURFACE* PRL_IO_SCREEN_SURFACE_PTR;
```

_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_DESKTOP_UTILITIES_STATE
Structure

Syntax

```
struct _PRL_IO_DESKTOP_UTILITIES_STATE {
  PRL_UINT32 bDesktopShown;
  PRL_UINT32 bTaskbarShown;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_GRACEFUL_SHUTDOWN Structure

Syntax

```
struct _PRL_IO_GRACEFUL_SHUTDOWN {
  PRL_UINT32 kind;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_KEYBOARD_LEDS Structure

Syntax

```
struct _PRL_IO_KEYBOARD_LEDS {
  PRL_UINT32 ledsState;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_KEYBOARD_SCANCODE Structure

Syntax

```
struct _PRL_IO_KEYBOARD_SCANCODE {
  PRL_UINT16 key;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_LANGUAGE_HOTKEYS Structure

Syntax

```
struct _PRL_IO_LANGUAGE_HOTKEYS {
  PRL_UINT32 LanguageHotkey;
  PRL_UINT32 LayoutHotkey;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR Structure

Syntax

```
struct _PRL_IO_MOUSE_CURSOR {
```

```
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 hotSpotX;
    PRL_UINT32 hotSpotY;
    PRL_UINT32 width;
    PRL_UINT32 height;
    struct _PRL_IO_SLIDING_MOUSE {
      PRL_UINT32 state;
    };
    struct _PRL_IO_AVAILABLE_DISPLAYS {
      PRL_UINT32 availableDisplays;
    };
    struct _PRL_IO_DEVICE_USED {
      PRL_UINT32 devType;
      PRL_UINT32 devIndex;
      PRL_UINT32 devAction;
    };
    struct _PRL_IO_GENERIC_BUFFER {
      PRL_UINT32 size;
      PRL_VOID_PTR data;
    };
    struct _PRL_TIS_VERSION {
      struct _PRL_TIS_INT_VERSION {
        struct _PRL_TIS_RECORD_INFO {
          struct _PRL_IO_TOOLS_UTILITY_COMMAND {
            struct _PRL_IO_TOOLS_SIA_DATA {
              struct _PRL_IO_DYNRES_TOOLS_STATUS {
                PRL_UINT32 status;
              };
              struct _PRL_PROXY_CLIENT_INFO {
                PRL_UINT32 structSize;
                PRL_CHAR clientOptions[64];
                PRL_UINT8 customerSupportUUID[16];
                PRL_CHAR managerURL[256];
              };
              struct _PRL_PROXY_REGISTRATION_INFO {
                PRL_CHAR hostAddresses[16][64];
                PRL_INT32 hostAddressesSz;
                PRL_CHAR hostName[64];
                PRL_UINT8 appUUID[16];
                PRL_CHAR appVersion[32];
                PRL_CHAR hostOptions[33];
                PRL_CHAR hostOsName[63];
                PRL_CHAR isoTimestamp[32];
              };
              struct _PRL_PROXY_HOST_INFO {
                PRL_UINT8 hostUUID[16];
                PRL_CHAR brokerAddresses[16][64];
                PRL_INT32 brokerAddressesSz;
                PRL_PROXY_REGISTRATION_INFO info;
              };
              struct _PRL_PROXY_GEO_INFO {
                PRL_CHAR countryName[64];
                PRL_CHAR countryCode[3];
                PRL_CHAR region[3];
                PRL_CHAR city[64];
                PRL_CHAR postalCode[10];
              };
              struct _PRL_PROXY_HOST_INFO_EXT {
                PRL_PROXY_HOST_INFO hostInfo;
                PRL_BOOL isOnline;
                PRL_PROXY_GEO_INFO_PTR geoInfo;
                struct _PRL_PROXY_HOST_INFO_EXT * next;
              };
              struct _PRL_PROXY_USER_DATA {
                PRL_STR dataKey;
```

```
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
    };
    struct _PRL_CPULIMIT_DATA {
      PRL_UINT32 value;
      PRL_CPULIMIT_TYPE type;
    };
    struct _PRL_IOLIMIT_DATA {
      PRL_UINT32 value;
      PRL_IOLIMIT_TYPE type;
    };
    struct _PRL_GET_VM_CONFIG_PARAM_DATA {
      PRL_VM_TYPE nVmType;
      PRL_CONST_STR sConfigSample;
      PRL_UINT32 nOsVersion;
    };
    struct _PRL_CVSRC_OPEN_MODE {
      PRL_UINT32 sessionId;
      PRL_UINT32 frameFormat;
      PRL_UINT32 frameWidth;
      PRL_UINT32 frameHeight;
```

```
                PRL_UINT32 frameBytesPerRow;
              };
              struct _PRL_IO_SSO_CLIENT_VM_PACKET {
                PRL_UINT8 credentialsRequestUUID[16];
                PRL_UINT32 reserved[12];
              };
              struct _PRL_IO_CHANNEL_STAT {
                struct _PRL_CPU_FEATURES_MASK {
                  struct _PRL_CPU_FEATURES {
                    struct _PRL_HOST_OS_INFO {
                      struct _PRL_MEMGUARANTEE_DATA {
                        PRL_UINT32 value;
                        PRL_MEMGUARANTEE_TYPE type;
                      };
                    };
                  };
                };
              };
            };
          };
        };
      };
    };
  };
};
```

File

PrlIOStructs.h

Members

_PRL_IO_MOUSE_CURSOR Data Members

	Name	Description
	height (p. 865)	
	hotSpotX (p. 865)	
	hotSpotY (p. 865)	
	width (p. 865)	
	x (p. 865)	
	y (p. 865)	

_PRL_IO_MOUSE_CURSOR Nested Types

Name	Description
PRL_IO_AVAILABLE_DISPLAYS (p. 866)	
PRL_IO_AVAILABLE_DISPLAYS_PTR (p. 866)	
PRL_IO_DEVICE_USED (p. 866)	
PRL_IO_DEVICE_USED_PTR (p. 866)	
PRL_IO_GENERIC_BUFFER (p. 866)	
PRL_IO_GENERIC_BUFFER_PTR (p. 866)	
PRL_IO_MOUSE_CURSOR_PTR (p. 866)	
PRL_IO_SLIDING_MOUSE (p. 867)	
PRL_IO_SLIDING_MOUSE_PTR (p. 867)	

_PRL_IO_MOUSE_CURSOR Structures

	Name	Description
	_PRL_IO_AVAILABLE_DISPLAYS (p. 867)	
	_PRL_IO_DEVICE_USED (p. 867)	
	_PRL_IO_GENERIC_BUFFER (p. 867)	
	_PRL_IO_SLIDING_MOUSE (p. 868)	
	_PRL_TIS_VERSION (p. 868)	@brief Structure that is used in TIS to represent versions * • TIS - Tools Information Service * • @sa PRL_TIS_INT_VERSION, PRL_TIS_RECORD_INFO

_PRL_IO_MOUSE_CURSOR Data Members

	Name	Description
	height (p. 865)	
	hotSpotX (p. 865)	
	hotSpotY (p. 865)	
	width (p. 865)	
	x (p. 865)	
	y (p. 865)	

_PRL_IO_MOUSE_CURSOR Nested Types

Name	Description
PRL_IO_AVAILABLE_DISPLAYS (p. 866)	
PRL_IO_AVAILABLE_DISPLAYS_PTR (p. 866)	
PRL_IO_DEVICE_USED (p. 866)	
PRL_IO_DEVICE_USED_PTR (p. 866)	
PRL_IO_GENERIC_BUFFER (p. 866)	
PRL_IO_GENERIC_BUFFER_PTR (p. 866)	
PRL_IO_MOUSE_CURSOR_PTR (p. 866)	
PRL_IO_SLIDING_MOUSE (p. 867)	
PRL_IO_SLIDING_MOUSE_PTR (p. 867)	

_PRL_IO_MOUSE_CURSOR Structures

	Name	Description
	_PRL_IO_AVAILABLE_DISPLAYS (p. 867)	
	_PRL_IO_DEVICE_USED (p. 867)	
	_PRL_IO_GENERIC_BUFFER (p. 867)	
	_PRL_IO_SLIDING_MOUSE (p. 868)	

	_PRL_TIS_VERSION (p. 868)	@brief Structure that is used in TIS to represent versions * • TIS - Tools Information Service * • @sa PRL_TIS_INT_VERSION, PRL_TIS_RECORD_INFO

_PRL_IO_MOUSE_CURSOR Data Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::height
Data Member

Syntax

```
PRL_UINT32 height;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::hotSpotX
Data Member

Syntax

```
PRL_UINT32 hotSpotX;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::hotSpotY
Data Member

Syntax

```
PRL_UINT32 hotSpotY;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::width
Data Member

Syntax

```
PRL_UINT32 width;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::x
Data Member

Syntax

```
PRL_UINT32 x;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::y
Data Member

Syntax

```
PRL_UINT32 y;
```

_PRL_IO_MOUSE_CURSOR Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_AVAILABLE_DISPLAYS Nested Type

Syntax

```
typedef struct _PRL_IO_AVAILABLE_DISPLAYS {
  PRL_UINT32 availableDisplays;
} PRL_IO_AVAILABLE_DISPLAYS;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_AVAILABLE_DISPLAYS_PTR Nested Type

Syntax

```
typedef PRL_IO_AVAILABLE_DISPLAYS* PRL_IO_AVAILABLE_DISPLAYS_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_DEVICE_USED Nested Type

Syntax

```
typedef struct _PRL_IO_DEVICE_USED {
  PRL_UINT32 devType;
  PRL_UINT32 devIndex;
  PRL_UINT32 devAction;
} PRL_IO_DEVICE_USED;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_DEVICE_USED_PTR Nested Type

Syntax

```
typedef PRL_IO_DEVICE_USED* PRL_IO_DEVICE_USED_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_GENERIC_BUFFER Nested Type

Syntax

```
typedef struct _PRL_IO_GENERIC_BUFFER {
  PRL_UINT32 size;
  PRL_VOID_PTR data;
} PRL_IO_GENERIC_BUFFER;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_GENERIC_BUFFER_PTR Nested Type

Syntax

```
typedef PRL_IO_GENERIC_BUFFER* PRL_IO_GENERIC_BUFFER_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_MOUSE_CURSOR_PTR Nested Type

Syntax

```
typedef PRL_IO_MOUSE_CURSOR* PRL_IO_MOUSE_CURSOR_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_SLIDING_MOUSE Nested Type

Syntax

```
typedef struct _PRL_IO_SLIDING_MOUSE {
  PRL_UINT32 state;
} PRL_IO_SLIDING_MOUSE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_SLIDING_MOUSE_PTR Nested Type

Syntax

```
typedef PRL_IO_SLIDING_MOUSE* PRL_IO_SLIDING_MOUSE_PTR;
```

_PRL_IO_MOUSE_CURSOR Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_AVAILABLE_DISPLAYS Structure

Syntax

```
struct _PRL_IO_AVAILABLE_DISPLAYS {
  PRL_UINT32 availableDisplays;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_DEVICE_USED Structure

Syntax

```
struct _PRL_IO_DEVICE_USED {
  PRL_UINT32 devType;
  PRL_UINT32 devIndex;
  PRL_UINT32 devAction;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_GENERIC_BUFFER Structure

Syntax

```
struct _PRL_IO_GENERIC_BUFFER {
  PRL_UINT32 size;
  PRL_VOID_PTR data;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_SLIDING_MOUSE Structure

Syntax

```
struct _PRL_IO_SLIDING_MOUSE {
  PRL_UINT32 state;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION Structure

@brief Structure that is used in TIS to represent versions *

- TIS - Tools Information Service

*

- @sa PRL_TIS_INT_VERSION, PRL_TIS_RECORD_INFO

Syntax

```
struct _PRL_TIS_VERSION {
  struct _PRL_TIS_INT_VERSION {
    struct _PRL_TIS_RECORD_INFO {
      struct _PRL_IO_TOOLS_UTILITY_COMMAND {
        struct _PRL_IO_TOOLS_SIA_DATA {
          struct _PRL_IO_DYNRES_TOOLS_STATUS {
            PRL_UINT32 status;
          };
          struct _PRL_PROXY_CLIENT_INFO {
            PRL_UINT32 structSize;
            PRL_CHAR clientOptions[64];
            PRL_UINT8 customerSupportUUID[16];
            PRL_CHAR managerURL[256];
          };
          struct _PRL_PROXY_REGISTRATION_INFO {
            PRL_CHAR hostAddresses[16][64];
            PRL_INT32 hostAddressesSz;
            PRL_CHAR hostName[64];
            PRL_UINT8 appUUID[16];
            PRL_CHAR appVersion[32];
            PRL_CHAR hostOptions[33];
            PRL_CHAR hostOsName[63];
            PRL_CHAR isoTimestamp[32];
          };
          struct _PRL_PROXY_HOST_INFO {
            PRL_UINT8 hostUUID[16];
            PRL_CHAR brokerAddresses[16][64];
            PRL_INT32 brokerAddressesSz;
            PRL_PROXY_REGISTRATION_INFO info;
          };
          struct _PRL_PROXY_GEO_INFO {
            PRL_CHAR countryName[64];
            PRL_CHAR countryCode[3];
            PRL_CHAR region[3];
            PRL_CHAR city[64];
            PRL_CHAR postalCode[10];
          };
          struct _PRL_PROXY_HOST_INFO_EXT {
```

```
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
    };
    struct _PRL_CPULIMIT_DATA {
      PRL_UINT32 value;
      PRL_CPULIMIT_TYPE type;
    };
    struct _PRL_IOLIMIT_DATA {
      PRL_UINT32 value;
      PRL_IOLIMIT_TYPE type;
    };
    struct _PRL_GET_VM_CONFIG_PARAM_DATA {
      PRL_VM_TYPE nVmType;
      PRL_CONST_STR sConfigSample;
```

```
                PRL_UINT32 nOsVersion;
              };
              struct _PRL_CVSRC_OPEN_MODE {
                PRL_UINT32 sessionId;
                PRL_UINT32 frameFormat;
                PRL_UINT32 frameWidth;
                PRL_UINT32 frameHeight;
                PRL_UINT32 frameBytesPerRow;
              };
              struct _PRL_IO_SSO_CLIENT_VM_PACKET {
                PRL_UINT8 credentialsRequestUUID[16];
                PRL_UINT32 reserved[12];
              };
              struct _PRL_IO_CHANNEL_STAT {
                struct _PRL_CPU_FEATURES_MASK {
                  struct _PRL_CPU_FEATURES {
                    struct _PRL_HOST_OS_INFO {
                      struct _PRL_MEMGUARANTEE_DATA {
                        PRL_UINT32 value;
                        PRL_MEMGUARANTEE_TYPE type;
                      };
                    };
                  };
                };
              };
            };
          };
        };
      };
    };
  };
};
```

File

PrlIOStructs.h

Members

_PRL_TIS_VERSION Nested Types

Name	Description
PRL_TIS_VERSION_PTR (p. 871)	

_PRL_TIS_VERSION Structures

	Name	Description
	_PRL_TIS_INT_VERSION (p. 871)	@brief Structure that is used in TIS to represent internal versions • TIS - Tools Information Service * • @sa PRL_TIS_VERSION, PRL_TIS_RECORD_INFO

_PRL_TIS_VERSION Nested Types

Name	Description
PRL_TIS_VERSION_PTR (p. 871)	

_PRL_TIS_VERSION Structures

	Name	Description
	_PRL_TIS_INT_VERSION (p. 871)	@brief Structure that is used in TIS to represent internal versions • TIS - Tools Information Service * • @sa PRL_TIS_VERSION, PRL_TIS_RECORD_INFO

_PRL_TIS_VERSION Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::PRL_TIS_VERSION_PTR
Nested Type

Syntax

```
typedef PRL_TIS_VERSION* PRL_TIS_VERSION_PTR;
```

_PRL_TIS_VERSION Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION
Structure

@brief Structure that is used in TIS to represent internal versions

- TIS - Tools Information Service

*

- @sa PRL_TIS_VERSION, PRL_TIS_RECORD_INFO

Syntax

```
struct _PRL_TIS_INT_VERSION {
  struct _PRL_TIS_RECORD_INFO {
    struct _PRL_IO_TOOLS_UTILITY_COMMAND {
      struct _PRL_IO_TOOLS_SIA_DATA {
        struct _PRL_IO_DYNRES_TOOLS_STATUS {
          PRL_UINT32 status;
        };
        struct _PRL_PROXY_CLIENT_INFO {
          PRL_UINT32 structSize;
          PRL_CHAR clientOptions[64];
          PRL_UINT8 customerSupportUUID[16];
          PRL_CHAR managerURL[256];
        };
        struct _PRL_PROXY_REGISTRATION_INFO {
          PRL_CHAR hostAddresses[16][64];
          PRL_INT32 hostAddressesSz;
          PRL_CHAR hostName[64];
          PRL_UINT8 appUUID[16];
          PRL_CHAR appVersion[32];
          PRL_CHAR hostOptions[33];
          PRL_CHAR hostOsName[63];
          PRL_CHAR isoTimestamp[32];
        };
        struct _PRL_PROXY_HOST_INFO {
          PRL_UINT8 hostUUID[16];
```

```
  PRL_CHAR brokerAddresses[16][64];
  PRL_INT32 brokerAddressesSz;
  PRL_PROXY_REGISTRATION_INFO info;
};
struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
```

```
                };
                struct _PRL_CPULIMIT_DATA {
                  PRL_UINT32 value;
                  PRL_CPULIMIT_TYPE type;
                };
                struct _PRL_IOLIMIT_DATA {
                  PRL_UINT32 value;
                  PRL_IOLIMIT_TYPE type;
                };
                struct _PRL_GET_VM_CONFIG_PARAM_DATA {
                  PRL_VM_TYPE nVmType;
                  PRL_CONST_STR sConfigSample;
                  PRL_UINT32 nOsVersion;
                };
                struct _PRL_CVSRC_OPEN_MODE {
                  PRL_UINT32 sessionId;
                  PRL_UINT32 frameFormat;
                  PRL_UINT32 frameWidth;
                  PRL_UINT32 frameHeight;
                  PRL_UINT32 frameBytesPerRow;
                };
                struct _PRL_IO_SSO_CLIENT_VM_PACKET {
                  PRL_UINT8 credentialsRequestUUID[16];
                  PRL_UINT32 reserved[12];
                };
                struct _PRL_IO_CHANNEL_STAT {
                  struct _PRL_CPU_FEATURES_MASK {
                    struct _PRL_CPU_FEATURES {
                      struct _PRL_HOST_OS_INFO {
                        struct _PRL_MEMGUARANTEE_DATA {
                          PRL_UINT32 value;
                          PRL_MEMGUARANTEE_TYPE type;
                        };
                      };
                    };
                  };
                };
              };
            };
          };
        };
      };
    };
  };
```

File

PrlIOStructs.h

Members

_PRL_TIS_INT_VERSION Nested Types

Name	Description
PRL_TIS_INT_VERSION_PTR (p. 874)	

_PRL_TIS_INT_VERSION Structures

	Name	Description
	_PRL_TIS_RECORD_INFO (p. 874)	@brief Fixed size information about TIS record • TIS - Tools Information Service * • @sa PRL_TIS_VERSION, PRL_TIS_INT_VERSION

_PRL_TIS_INT_VERSION Nested Types

Name	Description
PRL_TIS_INT_VERSION_PTR (p. 874)	

_PRL_TIS_INT_VERSION Structures

	Name	Description
	_PRL_TIS_RECORD_INFO (p. 874)	@brief Fixed size information about TIS record • TIS - Tools Information Service * • @sa PRL_TIS_VERSION, PRL_TIS_INT_VERSION

_PRL_TIS_INT_VERSION Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::PRL_TIS_INT_VERSION_PTR
Nested Type

Syntax

```
typedef PRL_TIS_INT_VERSION* PRL_TIS_INT_VERSION_PTR;
```

_PRL_TIS_INT_VERSION Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO
Structure

@brief Fixed size information about TIS record

- TIS - Tools Information Service

*

- @sa PRL_TIS_VERSION, PRL_TIS_INT_VERSION

Syntax

```
struct _PRL_TIS_RECORD_INFO {
  struct _PRL_IO_TOOLS_UTILITY_COMMAND {
    struct _PRL_IO_TOOLS_SIA_DATA {
      struct _PRL_IO_DYNRES_TOOLS_STATUS {
        PRL_UINT32 status;
      };
      struct _PRL_PROXY_CLIENT_INFO {
        PRL_UINT32 structSize;
        PRL_CHAR clientOptions[64];
        PRL_UINT8 customerSupportUUID[16];
        PRL_CHAR managerURL[256];
      };
      struct _PRL_PROXY_REGISTRATION_INFO {
        PRL_CHAR hostAddresses[16][64];
        PRL_INT32 hostAddressesSz;
        PRL_CHAR hostName[64];
        PRL_UINT8 appUUID[16];
        PRL_CHAR appVersion[32];
        PRL_CHAR hostOptions[33];
```

```
  PRL_CHAR hostOsName[63];
  PRL_CHAR isoTimestamp[32];
};
struct _PRL_PROXY_HOST_INFO {
  PRL_UINT8 hostUUID[16];
  PRL_CHAR brokerAddresses[16][64];
  PRL_INT32 brokerAddressesSz;
  PRL_PROXY_REGISTRATION_INFO info;
};
struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
```

```
            struct _PRL_STAT_NET_TRAFFIC {
              PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
              PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
              PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
              PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
            };
            struct _PRL_CPULIMIT_DATA {
              PRL_UINT32 value;
              PRL_CPULIMIT_TYPE type;
            };
            struct _PRL_IOLIMIT_DATA {
              PRL_UINT32 value;
              PRL_IOLIMIT_TYPE type;
            };
            struct _PRL_GET_VM_CONFIG_PARAM_DATA {
              PRL_VM_TYPE nVmType;
              PRL_CONST_STR sConfigSample;
              PRL_UINT32 nOsVersion;
            };
            struct _PRL_CVSRC_OPEN_MODE {
              PRL_UINT32 sessionId;
              PRL_UINT32 frameFormat;
              PRL_UINT32 frameWidth;
              PRL_UINT32 frameHeight;
              PRL_UINT32 frameBytesPerRow;
            };
            struct _PRL_IO_SSO_CLIENT_VM_PACKET {
              PRL_UINT8 credentialsRequestUUID[16];
              PRL_UINT32 reserved[12];
            };
            struct _PRL_IO_CHANNEL_STAT {
              struct _PRL_CPU_FEATURES_MASK {
                struct _PRL_CPU_FEATURES {
                  struct _PRL_HOST_OS_INFO {
                    struct _PRL_MEMGUARANTEE_DATA {
                      PRL_UINT32 value;
                      PRL_MEMGUARANTEE_TYPE type;
                    };
                  };
                };
              };
            };
          };
        };
      };
    };
  };
```

File

PrlIOStructs.h

Members

_PRL_TIS_RECORD_INFO Nested Types

Name	Description
PRL_TIS_RECORD_INFO_PTR (p. 877)	

_PRL_TIS_RECORD_INFO Structures

	Name	Description
	_PRL_IO_TOOLS_UTILITY_COMMAND (p. 877)	@brief Structure that represents Virtuozzo Tools Utility command • This structure is used to handle PET_IO_TOOLS_UTILITY_COMMAND.

_PRL_TIS_RECORD_INFO Nested Types

Name	Description
PRL_TIS_RECORD_INFO_PTR (p. 877)	

_PRL_TIS_RECORD_INFO Structures

	Name	Description
	_PRL_IO_TOOLS_UTILITY_COMMAND (p. 877)	@brief Structure that represents Virtuozzo Tools Utility command • This structure is used to handle PET_IO_TOOLS_UTILITY_COMMAND.

_PRL_TIS_RECORD_INFO Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::PRL_TIS_RECORD_INFO_PTR
Nested Type

Syntax

```
typedef PRL_TIS_RECORD_INFO* PRL_TIS_RECORD_INFO_PTR;
```

_PRL_TIS_RECORD_INFO Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND
Structure

@brief Structure that represents Virtuozzo Tools Utility command

- This structure is used to handle PET_IO_TOOLS_UTILITY_COMMAND.

Syntax

```
struct _PRL_IO_TOOLS_UTILITY_COMMAND {
  struct _PRL_IO_TOOLS_SIA_DATA {
    struct _PRL_IO_DYNRES_TOOLS_STATUS {
      PRL_UINT32 status;
    };
    struct _PRL_PROXY_CLIENT_INFO {
      PRL_UINT32 structSize;
      PRL_CHAR clientOptions[64];
      PRL_UINT8 customerSupportUUID[16];
      PRL_CHAR managerURL[256];
    };
    struct _PRL_PROXY_REGISTRATION_INFO {
      PRL_CHAR hostAddresses[16][64];
      PRL_INT32 hostAddressesSz;
      PRL_CHAR hostName[64];
      PRL_UINT8 appUUID[16];
      PRL_CHAR appVersion[32];
      PRL_CHAR hostOptions[33];
      PRL_CHAR hostOsName[63];
      PRL_CHAR isoTimestamp[32];
    };
    struct _PRL_PROXY_HOST_INFO {
      PRL_UINT8 hostUUID[16];
      PRL_CHAR brokerAddresses[16][64];
      PRL_INT32 brokerAddressesSz;
```

```
  PRL_PROXY_REGISTRATION_INFO info;
};
struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
    };
    struct _PRL_CPULIMIT_DATA {
```

```
            PRL_UINT32 value;
            PRL_CPULIMIT_TYPE type;
          };
          struct _PRL_IOLIMIT_DATA {
            PRL_UINT32 value;
            PRL_IOLIMIT_TYPE type;
          };
          struct _PRL_GET_VM_CONFIG_PARAM_DATA {
            PRL_VM_TYPE nVmType;
            PRL_CONST_STR sConfigSample;
            PRL_UINT32 nOsVersion;
          };
          struct _PRL_CVSRC_OPEN_MODE {
            PRL_UINT32 sessionId;
            PRL_UINT32 frameFormat;
            PRL_UINT32 frameWidth;
            PRL_UINT32 frameHeight;
            PRL_UINT32 frameBytesPerRow;
          };
          struct _PRL_IO_SSO_CLIENT_VM_PACKET {
            PRL_UINT8 credentialsRequestUUID[16];
            PRL_UINT32 reserved[12];
          };
          struct _PRL_IO_CHANNEL_STAT {
            struct _PRL_CPU_FEATURES_MASK {
              struct _PRL_CPU_FEATURES {
                struct _PRL_HOST_OS_INFO {
                  struct _PRL_MEMGUARANTEE_DATA {
                    PRL_UINT32 value;
                    PRL_MEMGUARANTEE_TYPE type;
                  };
                };
              };
            };
          };
        };
      };
    };
  };
```

File

PrlIOStructs.h

Members

_PRL_IO_TOOLS_UTILITY_COMMAND Nested Types

Name	Description
PRL_IO_TOOLS_UTILITY_COMMAND_PTR (p. 880)	

_PRL_IO_TOOLS_UTILITY_COMMAND Structures

	Name	Description
	_PRL_IO_TOOLS_SIA_DATA (p. 880)	@brief Structure that represents Virtuozzo Tools Shared Internet Applications data • This structure is used to handle message of PET_IO_TOOLS_SIA_DATA type.

_PRL_IO_TOOLS_UTILITY_COMMAND Nested Types

Name	Description
PRL_IO_TOOLS_UTILITY_COMMAND_PTR (p. 880)	

_PRL_IO_TOOLS_UTILITY_COMMAND Structures

	Name	Description
	_PRL_IO_TOOLS_SIA_DATA (p. 880)	@brief Structure that represents Virtuozzo Tools Shared Internet Applications data • This structure is used to handle message of PET_IO_TOOLS_SIA_DATA type.

_PRL_IO_TOOLS_UTILITY_COMMAND Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::PRL_IO_TOOLS_UTILITY_COMMAND_PTR Nested Type

Syntax

```
typedef PRL_IO_TOOLS_UTILITY_COMMAND* PRL_IO_TOOLS_UTILITY_COMMAND_PTR;
```

_PRL_IO_TOOLS_UTILITY_COMMAND Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA Structure

@brief Structure that represents Virtuozzo Tools Shared Internet Applications data

- This structure is used to handle message of PET_IO_TOOLS_SIA_DATA type.

Syntax

```
struct _PRL_IO_TOOLS_SIA_DATA {
  struct _PRL_IO_DYNRES_TOOLS_STATUS {
    PRL_UINT32 status;
  };
  struct _PRL_PROXY_CLIENT_INFO {
    PRL_UINT32 structSize;
    PRL_CHAR clientOptions[64];
    PRL_UINT8 customerSupportUUID[16];
    PRL_CHAR managerURL[256];
  };
  struct _PRL_PROXY_REGISTRATION_INFO {
    PRL_CHAR hostAddresses[16][64];
    PRL_INT32 hostAddressesSz;
    PRL_CHAR hostName[64];
    PRL_UINT8 appUUID[16];
    PRL_CHAR appVersion[32];
    PRL_CHAR hostOptions[33];
    PRL_CHAR hostOsName[63];
    PRL_CHAR isoTimestamp[32];
  };
  struct _PRL_PROXY_HOST_INFO {
    PRL_UINT8 hostUUID[16];
    PRL_CHAR brokerAddresses[16][64];
    PRL_INT32 brokerAddressesSz;
    PRL_PROXY_REGISTRATION_INFO info;
  };
  struct _PRL_PROXY_GEO_INFO {
    PRL_CHAR countryName[64];
```

```
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
};
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
  struct _PRL_IO_SCREEN_UPDATE {
    PRL_UINT32 offset;
    PRL_UINT32 x;
    PRL_UINT32 y;
    PRL_UINT32 width;
    PRL_UINT32 height;
    PRL_UINT32 display;
    struct _PRL_STAT_NET_TRAFFIC {
      PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
      PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
      PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
      PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
    };
    struct _PRL_CPULIMIT_DATA {
      PRL_UINT32 value;
      PRL_CPULIMIT_TYPE type;
    };
    struct _PRL_IOLIMIT_DATA {
```

```
          PRL_UINT32 value;
          PRL_IOLIMIT_TYPE type;
        };
        struct _PRL_GET_VM_CONFIG_PARAM_DATA {
          PRL_VM_TYPE nVmType;
          PRL_CONST_STR sConfigSample;
          PRL_UINT32 nOsVersion;
        };
        struct _PRL_CVSRC_OPEN_MODE {
          PRL_UINT32 sessionId;
          PRL_UINT32 frameFormat;
          PRL_UINT32 frameWidth;
          PRL_UINT32 frameHeight;
          PRL_UINT32 frameBytesPerRow;
        };
        struct _PRL_IO_SSO_CLIENT_VM_PACKET {
          PRL_UINT8 credentialsRequestUUID[16];
          PRL_UINT32 reserved[12];
        };
        struct _PRL_IO_CHANNEL_STAT {
          struct _PRL_CPU_FEATURES_MASK {
            struct _PRL_CPU_FEATURES {
              struct _PRL_HOST_OS_INFO {
                struct _PRL_MEMGUARANTEE_DATA {
                  PRL_UINT32 value;
                  PRL_MEMGUARANTEE_TYPE type;
                };
              };
            };
          };
        };
      };
    };
  };
```

File

PrlIOStructs.h

Members

_PRL_IO_TOOLS_SIA_DATA Nested Types

Name	Description
PRL_IO_DYNRES_TOOLS_STATUS (p. 884)	
PRL_IO_DYNRES_TOOLS_STATUS_PTR (p. 884)	
PRL_IO_TOOLS_SIA_DATA_PTR (p. 884)	
PRL_PROXY_CLIENT_INFO (p. 885)	
PRL_PROXY_CLIENT_INFO_PTR (p. 885)	
PRL_PROXY_DATA_KEY (p. 885)	
PRL_PROXY_DATA_KEY_PTR (p. 886)	
PRL_PROXY_GEO_INFO (p. 886)	
PRL_PROXY_GEO_INFO_PTR (p. 886)	
PRL_PROXY_HOST_INFO (p. 886)	
PRL_PROXY_HOST_INFO_EXT (p. 887)	
PRL_PROXY_HOST_INFO_EXT_PTR (p. 887)	
PRL_PROXY_HOST_INFO_PTR (p. 887)	
PRL_PROXY_REGISTRATION_INFO (p. 887)	

PRL_PROXY_REGISTRATION_INFO_PTR (p. 888)	
PRL_PROXY_USER_DATA (p. 888)	
PRL_PROXY_USER_DATA_PTR (p. 889)	

_PRL_IO_TOOLS_SIA_DATA Structures

	Name	Description
	_PRL_IO_DYNRES_TOOLS_STATUS (p. 889)	
	_PRL_IO_MEMORY_INFO (p. 889)	
	_PRL_PROXY_CLIENT_INFO (p. 911)	
	_PRL_PROXY_DATA_KEY (p. 912)	
	_PRL_PROXY_GEO_INFO (p. 912)	
	_PRL_PROXY_HOST_INFO (p. 912)	
	_PRL_PROXY_HOST_INFO_EXT (p. 913)	
	_PRL_PROXY_REGISTRATION_INFO (p. 913)	
	_PRL_PROXY_USER_DATA (p. 914)	

_PRL_IO_TOOLS_SIA_DATA Nested Types

Name	Description
PRL_IO_DYNRES_TOOLS_STATUS (p. 884)	
PRL_IO_DYNRES_TOOLS_STATUS_PTR (p. 884)	
PRL_IO_TOOLS_SIA_DATA_PTR (p. 884)	
PRL_PROXY_CLIENT_INFO (p. 885)	
PRL_PROXY_CLIENT_INFO_PTR (p. 885)	
PRL_PROXY_DATA_KEY (p. 885)	
PRL_PROXY_DATA_KEY_PTR (p. 886)	
PRL_PROXY_GEO_INFO (p. 886)	
PRL_PROXY_GEO_INFO_PTR (p. 886)	
PRL_PROXY_HOST_INFO (p. 886)	
PRL_PROXY_HOST_INFO_EXT (p. 887)	
PRL_PROXY_HOST_INFO_EXT_PTR (p. 887)	
PRL_PROXY_HOST_INFO_PTR (p. 887)	
PRL_PROXY_REGISTRATION_INFO (p. 887)	
PRL_PROXY_REGISTRATION_INFO_PTR (p. 888)	
PRL_PROXY_USER_DATA (p. 888)	
PRL_PROXY_USER_DATA_PTR (p. 889)	

_PRL_IO_TOOLS_SIA_DATA Structures

	Name	Description
	_PRL_IO_DYNRES_TOOLS_STATUS (p. 889)	
	_PRL_IO_MEMORY_INFO (p. 889)	

	_PRL_PROXY_CLIENT_INFO (p. 911)	
	_PRL_PROXY_DATA_KEY (p. 912)	
	_PRL_PROXY_GEO_INFO (p. 912)	
	_PRL_PROXY_HOST_INFO (p. 912)	
	_PRL_PROXY_HOST_INFO_EXT (p. 913)	
	_PRL_PROXY_REGISTRATION_INFO (p. 913)	
	_PRL_PROXY_USER_DATA (p. 914)	

_PRL_IO_TOOLS_SIA_DATA Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_IO_DYNRES_TOOLS_STATUS
Nested Type

Syntax

```
typedef struct _PRL_IO_DYNRES_TOOLS_STATUS {
  PRL_UINT32 status;
} PRL_IO_DYNRES_TOOLS_STATUS;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_IO_DYNRES_TOOLS_STATUS_PTR
Nested Type

Syntax

```
typedef PRL_IO_DYNRES_TOOLS_STATUS* PRL_IO_DYNRES_TOOLS_STATUS_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_IO_TOOLS_SIA_DATA_PTR
Nested Type

Syntax

```
typedef PRL_IO_TOOLS_SIA_DATA* PRL_IO_TOOLS_SIA_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_CLIENT_INFO
Nested Type

Syntax

```
typedef struct _PRL_PROXY_CLIENT_INFO {
  PRL_UINT32 structSize;
  PRL_CHAR clientOptions[64];
  PRL_UINT8 customerSupportUUID[16];
  PRL_CHAR managerURL[256];
} PRL_PROXY_CLIENT_INFO;
```

Members

structSize
< Should be filled by the caller with actual structure size

clientOptions
< Reserved

customerSupportUUID
< Reserved

managerURL
< Reserved

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_CLIENT_INFO_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_CLIENT_INFO* PRL_PROXY_CLIENT_INFO_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_DATA_KEY
Nested Type

Syntax

```
typedef struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
} PRL_PROXY_DATA_KEY;
```

Members

dataKey
< Data identifier

dataVersion
< Data version

next
< Pointer to the next element

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_DATA_KEY_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_DATA_KEY* PRL_PROXY_DATA_KEY_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_GEO_INFO
Nested Type

Syntax

```
typedef struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
} PRL_PROXY_GEO_INFO;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_GEO_INFO_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_GEO_INFO* PRL_PROXY_GEO_INFO_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO
Nested Type

Syntax

```
typedef struct _PRL_PROXY_HOST_INFO {
  PRL_UINT8 hostUUID[16];
  PRL_CHAR brokerAddresses[16][64];
  PRL_INT32 brokerAddressesSz;
  PRL_PROXY_REGISTRATION_INFO info;
} PRL_PROXY_HOST_INFO;
```

Members

hostUUID

< Proxy host uuid

brokerAddresses

< Broker addresses to which connection should be established

brokerAddressesSz

< Broker addresses size

info

< Host registration info

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO_EXT
Nested Type

Syntax

```
typedef struct _PRL_PROXY_HOST_INFO_EXT {
  PRL_PROXY_HOST_INFO hostInfo;
  PRL_BOOL isOnline;
  PRL_PROXY_GEO_INFO_PTR geoInfo;
  struct _PRL_PROXY_HOST_INFO_EXT * next;
} PRL_PROXY_HOST_INFO_EXT;
```

Members

isOnline

Sign which specifies whether appropriate host online

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO_EXT_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_HOST_INFO_EXT* PRL_PROXY_HOST_INFO_EXT_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_HOST_INFO* PRL_PROXY_HOST_INFO_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_REGISTRATION_INFO
Nested Type

Syntax

```
typedef struct _PRL_PROXY_REGISTRATION_INFO {
  PRL_CHAR hostAddresses[16][64];
  PRL_INT32 hostAddressesSz;
  PRL_CHAR hostName[64];
  PRL_UINT8 appUUID[16];
  PRL_CHAR appVersion[32];
  PRL_CHAR hostOptions[33];
  PRL_CHAR hostOsName[63];
  PRL_CHAR isoTimestamp[32];
} PRL_PROXY_REGISTRATION_INFO;
```

Members

hostAddresses

< Host addresses which host listens to

hostAddressesSz
< Size of host addresses

hostName
< Short host name

appUUID
< Application uuid

appVersion
< Application version

hostOptions
< Using for host icon hash transport

hostOsName
< Reserved

isoTimestamp
< Reserved

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_REGISTRATION_INFO_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_REGISTRATION_INFO* PRL_PROXY_REGISTRATION_INFO_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_USER_DATA
Nested Type

Syntax

```
typedef struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
} PRL_PROXY_USER_DATA;
```

Members

dataKey
< Data identifier

dataVersion
< Data version

data
< Pointer to the data buffer

dataSize
< Data size

next

< Pointer to the next element

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_USER_DATA_PTR
Nested Type

Syntax

```
typedef PRL_PROXY_USER_DATA* PRL_PROXY_USER_DATA_PTR;
```

_PRL_IO_TOOLS_SIA_DATA Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_DYNRES_TOOLS_STATUS
Structure

Syntax

```
struct _PRL_IO_DYNRES_TOOLS_STATUS {
  PRL_UINT32 status;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO
Structure

Syntax

```
struct _PRL_IO_MEMORY_INFO {
  PRL_UINT64 ramOffset;
  PRL_UINT64 ramSize;
  PRL_UINT64 vramOffset;
  PRL_UINT64 vramSize;
  PRL_UINT32 filenameSize;
  struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  };
  struct _PRL_IO_AUTH_RESPONSE {
    PRL_UINT32 authResult;
  };
  struct _PRL_IO_ATTACH_REQUEST {
    PRL_UINT32 isClientLocal;
    PRL_UINT32 encodingType;
    PRL_UINT8 vmUuid[16];
    PRL_UINT32 attachFlags;
  };
  struct _PRL_IO_ATTACH_RESPONSE {
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  };
```

```
    struct _PRL_IO_SCREEN_UPDATE {
      PRL_UINT32 offset;
      PRL_UINT32 x;
      PRL_UINT32 y;
      PRL_UINT32 width;
      PRL_UINT32 height;
      PRL_UINT32 display;
      struct _PRL_STAT_NET_TRAFFIC {
        PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
        PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
        PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
        PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
      };
      struct _PRL_CPULIMIT_DATA {
        PRL_UINT32 value;
        PRL_CPULIMIT_TYPE type;
      };
      struct _PRL_IOLIMIT_DATA {
        PRL_UINT32 value;
        PRL_IOLIMIT_TYPE type;
      };
      struct _PRL_GET_VM_CONFIG_PARAM_DATA {
        PRL_VM_TYPE nVmType;
        PRL_CONST_STR sConfigSample;
        PRL_UINT32 nOsVersion;
      };
      struct _PRL_CVSRC_OPEN_MODE {
        PRL_UINT32 sessionId;
        PRL_UINT32 frameFormat;
        PRL_UINT32 frameWidth;
        PRL_UINT32 frameHeight;
        PRL_UINT32 frameBytesPerRow;
      };
      struct _PRL_IO_SSO_CLIENT_VM_PACKET {
        PRL_UINT8 credentialsRequestUUID[16];
        PRL_UINT32 reserved[12];
      };
      struct _PRL_IO_CHANNEL_STAT {
        struct _PRL_CPU_FEATURES_MASK {
          struct _PRL_CPU_FEATURES {
            struct _PRL_HOST_OS_INFO {
              struct _PRL_MEMGUARANTEE_DATA {
                PRL_UINT32 value;
                PRL_MEMGUARANTEE_TYPE type;
              };
            };
          };
        };
      };
    };
};
```

File

PrlIOStructs.h

Members

_PRL_IO_MEMORY_INFO Data Members

	Name	Description
	filenameSize (p. 892)	
	ramOffset (p. 892)	
	ramSize (p. 892)	
	vramOffset (p. 893)	

	vramSize (p. 893)	

_PRL_IO_MEMORY_INFO Nested Types

Name	Description
PRL_IO_ATTACH_REQUEST (p. 893)	
PRL_IO_ATTACH_REQUEST_PTR (p. 893)	
PRL_IO_ATTACH_RESPONSE (p. 893)	
PRL_IO_ATTACH_RESPONSE_PTR (p. 894)	
PRL_IO_AUTH_EXEC_REQUEST (p. 894)	
PRL_IO_AUTH_EXEC_REQUEST_PTR (p. 894)	
PRL_IO_AUTH_REQUEST (p. 894)	
PRL_IO_AUTH_REQUEST_PTR (p. 894)	
PRL_IO_AUTH_RESPONSE (p. 895)	
PRL_IO_AUTH_RESPONSE_PTR (p. 895)	
PRL_IO_MEMORY_INFO_PTR (p. 895)	

_PRL_IO_MEMORY_INFO Structures

	Name	Description
	_PRL_IO_ATTACH_REQUEST (p. 895)	
	_PRL_IO_ATTACH_RESPONSE (p. 896)	
	_PRL_IO_AUTH_EXEC_REQUEST (p. 896)	
	_PRL_IO_AUTH_REQUEST (p. 896)	
	_PRL_IO_AUTH_RESPONSE (p. 896)	
	_PRL_IO_SCREEN_UPDATE (p. 897)	

_PRL_IO_MEMORY_INFO Data Members

	Name	Description
	filenameSize (p. 892)	
	ramOffset (p. 892)	
	ramSize (p. 892)	
	vramOffset (p. 893)	
	vramSize (p. 893)	

_PRL_IO_MEMORY_INFO Nested Types

Name	Description
PRL_IO_ATTACH_REQUEST (p. 893)	
PRL_IO_ATTACH_REQUEST_PTR (p. 893)	
PRL_IO_ATTACH_RESPONSE (p. 893)	
PRL_IO_ATTACH_RESPONSE_PTR (p. 894)	
PRL_IO_AUTH_EXEC_REQUEST (p. 894)	
PRL_IO_AUTH_EXEC_REQUEST_PTR (p. 894)	
PRL_IO_AUTH_REQUEST (p. 894)	

PRL_IO_AUTH_REQUEST_PTR (p. 894)	
PRL_IO_AUTH_RESPONSE (p. 895)	
PRL_IO_AUTH_RESPONSE_PTR (p. 895)	
PRL_IO_MEMORY_INFO_PTR (p. 895)	

_PRL_IO_MEMORY_INFO Structures

	Name	Description
	_PRL_IO_ATTACH_REQUEST (p. 895)	
	_PRL_IO_ATTACH_RESPONSE (p. 896)	
	_PRL_IO_AUTH_EXEC_REQUEST (p. 896)	
	_PRL_IO_AUTH_REQUEST (p. 896)	
	_PRL_IO_AUTH_RESPONSE (p. 896)	
	_PRL_IO_SCREEN_UPDATE (p. 897)	

_PRL_IO_MEMORY_INFO Data Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::filenameSize
Data Member

Syntax

```
PRL_UINT32 filenameSize;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::ramOffset
Data Member

Syntax

```
PRL_UINT64 ramOffset;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::ramSize
Data Member

Syntax

```
PRL_UINT64 ramSize;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::vramOffset
Data Member

Syntax

```
PRL_UINT64 vramOffset;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::vramSize
Data Member

Syntax

```
PRL_UINT64 vramSize;
```

_PRL_IO_MEMORY_INFO Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_REQUEST
Nested Type

Syntax

```
typedef struct _PRL_IO_ATTACH_REQUEST {
  PRL_UINT32 isClientLocal;
  PRL_UINT32 encodingType;
  PRL_UINT8 vmUuid[16];
  PRL_UINT32 attachFlags;
} PRL_IO_ATTACH_REQUEST;
```

Members

attachFlags

Since 6.4 IO protocol version

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_REQUEST_PTR
Nested Type

Syntax

```
typedef PRL_IO_ATTACH_REQUEST* PRL_IO_ATTACH_REQUEST_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_RESPONSE
Nested Type

Syntax

```
typedef struct _PRL_IO_ATTACH_RESPONSE {
```

```
    PRL_UINT32 attachResult;
    PRL_UINT32 encodingResult;
  } PRL_IO_ATTACH_RESPONSE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_RESPONSE_PTR
Nested Type

Syntax

```
  typedef PRL_IO_ATTACH_RESPONSE* PRL_IO_ATTACH_RESPONSE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_EXEC_REQUEST
Nested Type

Syntax

```
  typedef struct _PRL_IO_AUTH_EXEC_REQUEST {
    PRL_UINT8 sessionUuid[16];
  } PRL_IO_AUTH_EXEC_REQUEST;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_EXEC_REQUEST_PTR
Nested Type

Syntax

```
  typedef PRL_IO_AUTH_EXEC_REQUEST* PRL_IO_AUTH_EXEC_REQUEST_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_REQUEST
Nested Type

Syntax

```
  typedef struct _PRL_IO_AUTH_REQUEST {
    PRL_UINT8 sessionUuid[16];
  } PRL_IO_AUTH_REQUEST;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_REQUEST_PTR
Nested Type

Syntax

```
  typedef PRL_IO_AUTH_REQUEST* PRL_IO_AUTH_REQUEST_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_RESPONSE
Nested Type

Syntax

```
typedef struct _PRL_IO_AUTH_RESPONSE {
  PRL_UINT32 authResult;
} PRL_IO_AUTH_RESPONSE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_RESPONSE_PTR
Nested Type

Syntax

```
typedef PRL_IO_AUTH_RESPONSE* PRL_IO_AUTH_RESPONSE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_MEMORY_INFO_PTR
Nested Type

Syntax

```
typedef PRL_IO_MEMORY_INFO* PRL_IO_MEMORY_INFO_PTR;
```

_PRL_IO_MEMORY_INFO Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_ATTACH_REQUEST
Structure

Syntax

```
struct _PRL_IO_ATTACH_REQUEST {
  PRL_UINT32 isClientLocal;
  PRL_UINT32 encodingType;
  PRL_UINT8 vmUuid[16];
  PRL_UINT32 attachFlags;
};
```

File

PrlIOStructs.h

Members

attachFlags

Since 6.4 IO protocol version

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_ATTACH_RESPONSE
Structure

Syntax

```
struct _PRL_IO_ATTACH_RESPONSE {
  PRL_UINT32 attachResult;
  PRL_UINT32 encodingResult;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_EXEC_REQUEST
Structure

Syntax

```
struct _PRL_IO_AUTH_EXEC_REQUEST {
  PRL_UINT8 sessionUuid[16];
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_REQUEST
Structure

Syntax

```
struct _PRL_IO_AUTH_REQUEST {
  PRL_UINT8 sessionUuid[16];
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_RESPONSE
Structure

Syntax

```
struct _PRL_IO_AUTH_RESPONSE {
  PRL_UINT32 authResult;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE
Structure

Syntax

```
struct _PRL_IO_SCREEN_UPDATE {
  PRL_UINT32 offset;
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
  struct _PRL_STAT_NET_TRAFFIC {
    PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
    PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
    PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
    PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
  };
  struct _PRL_CPULIMIT_DATA {
    PRL_UINT32 value;
    PRL_CPULIMIT_TYPE type;
  };
  struct _PRL_IOLIMIT_DATA {
    PRL_UINT32 value;
    PRL_IOLIMIT_TYPE type;
  };
  struct _PRL_GET_VM_CONFIG_PARAM_DATA {
    PRL_VM_TYPE nVmType;
    PRL_CONST_STR sConfigSample;
    PRL_UINT32 nOsVersion;
  };
  struct _PRL_CVSRC_OPEN_MODE {
    PRL_UINT32 sessionId;
    PRL_UINT32 frameFormat;
    PRL_UINT32 frameWidth;
    PRL_UINT32 frameHeight;
    PRL_UINT32 frameBytesPerRow;
  };
  struct _PRL_IO_SSO_CLIENT_VM_PACKET {
    PRL_UINT8 credentialsRequestUUID[16];
    PRL_UINT32 reserved[12];
  };
  struct _PRL_IO_CHANNEL_STAT {
    struct _PRL_CPU_FEATURES_MASK {
      struct _PRL_CPU_FEATURES {
        struct _PRL_HOST_OS_INFO {
          struct _PRL_MEMGUARANTEE_DATA {
            PRL_UINT32 value;
            PRL_MEMGUARANTEE_TYPE type;
          };
        };
      };
    };
  };
};
```

File

PrlIOStructs.h

Members

_PRL_IO_SCREEN_UPDATE Data Members

	Name	Description
	display (p. 900)	
	height (p. 900)	
	offset (p. 900)	
	width (p. 900)	
	x (p. 900)	
	y (p. 900)	

_PRL_IO_SCREEN_UPDATE Nested Types

Name	Description
PRL_CONST_CPULIMIT_DATA_PTR (p. 901)	
PRL_CONST_CVSRC_OPEN_MODE (p. 901)	
PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR (p. 901)	
PRL_CONST_IOLIMIT_DATA_PTR (p. 901)	
PRL_CPULIMIT_DATA (p. 901)	
PRL_CPULIMIT_DATA_PTR (p. 902)	
PRL_CVSRC_OPEN_MODE (p. 902)	
PRL_CVSRC_OPEN_MODE_PTR (p. 902)	
PRL_GET_VM_CONFIG_PARAM_DATA (p. 902)	
PRL_GET_VM_CONFIG_PARAM_DATA_PTR (p. 902)	
PRL_IO_SCREEN_UPDATE_PTR (p. 903)	
PRL_IO_SSO_CLIENT_VM_PACKET (p. 903)	
PRL_IO_SSO_CLIENT_VM_PACKET_PTR (p. 903)	
PRL_IOLIMIT_DATA (p. 903)	
PRL_IOLIMIT_DATA_PTR (p. 903)	
PRL_STAT_NET_TRAFFIC (p. 904)	
PRL_STAT_NET_TRAFFIC_PTR (p. 904)	

_PRL_IO_SCREEN_UPDATE Structures

	Name	Description
	_PRL_CPULIMIT_DATA (p. 904)	
	_PRL_CVSRC_OPEN_MODE (p. 904)	
	_PRL_GET_VM_CONFIG_PARAM_DATA (p. 905)	
	_PRL_IO_CHANNEL_STAT (p. 905)	
	_PRL_IO_SSO_CLIENT_VM_PACKET (p. 910)	
	_PRL_IOLIMIT_DATA (p. 911)	
	_PRL_STAT_NET_TRAFFIC (p. 911)	

_PRL_IO_SCREEN_UPDATE Data Members

	Name	Description
	display (p. 900)	
	height (p. 900)	
	offset (p. 900)	
	width (p. 900)	
	x (p. 900)	
	y (p. 900)	

_PRL_IO_SCREEN_UPDATE Nested Types

Name	Description
PRL_CONST_CPULIMIT_DATA_PTR (p. 901)	
PRL_CONST_CVSRC_OPEN_MODE (p. 901)	
PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR (p. 901)	
PRL_CONST_IOLIMIT_DATA_PTR (p. 901)	
PRL_CPULIMIT_DATA (p. 901)	
PRL_CPULIMIT_DATA_PTR (p. 902)	
PRL_CVSRC_OPEN_MODE (p. 902)	
PRL_CVSRC_OPEN_MODE_PTR (p. 902)	
PRL_GET_VM_CONFIG_PARAM_DATA (p. 902)	
PRL_GET_VM_CONFIG_PARAM_DATA_PTR (p. 902)	
PRL_IO_SCREEN_UPDATE_PTR (p. 903)	
PRL_IO_SSO_CLIENT_VM_PACKET (p. 903)	
PRL_IO_SSO_CLIENT_VM_PACKET_PTR (p. 903)	
PRL_IOLIMIT_DATA (p. 903)	
PRL_IOLIMIT_DATA_PTR (p. 903)	
PRL_STAT_NET_TRAFFIC (p. 904)	
PRL_STAT_NET_TRAFFIC_PTR (p. 904)	

_PRL_IO_SCREEN_UPDATE Structures

	Name	Description
	_PRL_CPULIMIT_DATA (p. 904)	
	_PRL_CVSRC_OPEN_MODE (p. 904)	
	_PRL_GET_VM_CONFIG_PARAM_DATA (p. 905)	
	_PRL_IO_CHANNEL_STAT (p. 905)	
	_PRL_IO_SSO_CLIENT_VM_PACKET (p. 910)	
	_PRL_IOLIMIT_DATA (p. 911)	
	_PRL_STAT_NET_TRAFFIC (p. 911)	

_PRL_IO_SCREEN_UPDATE Data Members

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::display
Data Member

Syntax

```
PRL_UINT32 display;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::height
Data Member

Syntax

```
PRL_UINT32 height;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::offset
Data Member

Syntax

```
PRL_UINT32 offset;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::width
Data Member

Syntax

```
PRL_UINT32 width;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::x
Data Member

Syntax

```
PRL_UINT32 x;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::y
Data Member

Syntax

```
PRL_UINT32 y;
```

_PRL_IO_SCREEN_UPDATE Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_CPULIMIT_DATA_PTR
Nested Type

Syntax

```
typedef const PRL_CPULIMIT_DATA* PRL_CONST_CPULIMIT_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_CVSRC_OPEN_MODE
Nested Type

Syntax

```
typedef const PRL_CVSRC_OPEN_MODE* PRL_CONST_CVSRC_OPEN_MODE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR
Nested Type

Syntax

```
typedef const PRL_GET_VM_CONFIG_PARAM_DATA* PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_IOLIMIT_DATA_PTR
Nested Type

Syntax

```
typedef const PRL_IOLIMIT_DATA* PRL_CONST_IOLIMIT_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CPULIMIT_DATA
Nested Type

Syntax

```
typedef struct _PRL_CPULIMIT_DATA {
  PRL_UINT32 value;
  PRL_CPULIMIT_TYPE type;
} PRL_CPULIMIT_DATA;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CPULIMIT_DATA_PTR
Nested Type

Syntax

```
typedef PRL_CPULIMIT_DATA* PRL_CPULIMIT_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CVSRC_OPEN_MODE
Nested Type

Syntax

```
typedef struct _PRL_CVSRC_OPEN_MODE {
  PRL_UINT32 sessionId;
  PRL_UINT32 frameFormat;
  PRL_UINT32 frameWidth;
  PRL_UINT32 frameHeight;
  PRL_UINT32 frameBytesPerRow;
} PRL_CVSRC_OPEN_MODE;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CVSRC_OPEN_MODE_PTR
Nested Type

Syntax

```
typedef PRL_CVSRC_OPEN_MODE* PRL_CVSRC_OPEN_MODE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_GET_VM_CONFIG_PARAM_DATA
Nested Type

Syntax

```
typedef struct _PRL_GET_VM_CONFIG_PARAM_DATA {
  PRL_VM_TYPE nVmType;
  PRL_CONST_STR sConfigSample;
  PRL_UINT32 nOsVersion;
} PRL_GET_VM_CONFIG_PARAM_DATA;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_GET_VM_CONFIG_PARAM_DATA_PTR
Nested Type

Syntax

```
typedef PRL_GET_VM_CONFIG_PARAM_DATA* PRL_GET_VM_CONFIG_PARAM_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IO_SCREEN_UPDATE_PTR
Nested Type

Syntax

```
typedef PRL_IO_SCREEN_UPDATE* PRL_IO_SCREEN_UPDATE_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IO_SSO_CLIENT_VM_PACKET
Nested Type

Syntax

```
typedef struct _PRL_IO_SSO_CLIENT_VM_PACKET {
  PRL_UINT8 credentialsRequestUUID[16];
  PRL_UINT32 reserved[12];
} PRL_IO_SSO_CLIENT_VM_PACKET;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IO_SSO_CLIENT_VM_PACKET_PTR
Nested Type

Syntax

```
typedef PRL_IO_SSO_CLIENT_VM_PACKET* PRL_IO_SSO_CLIENT_VM_PACKET_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IOLIMIT_DATA
Nested Type

Syntax

```
typedef struct _PRL_IOLIMIT_DATA {
  PRL_UINT32 value;
  PRL_IOLIMIT_TYPE type;
} PRL_IOLIMIT_DATA;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IOLIMIT_DATA_PTR
Nested Type

Syntax

```
typedef PRL_IOLIMIT_DATA* PRL_IOLIMIT_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_STAT_NET_TRAFFIC
Nested Type

Syntax

```
typedef struct _PRL_STAT_NET_TRAFFIC {
  PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
  PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
  PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
  PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
} PRL_STAT_NET_TRAFFIC;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_STAT_NET_TRAFFIC_PTR
Nested Type

Syntax

```
typedef PRL_STAT_NET_TRAFFIC* PRL_STAT_NET_TRAFFIC_PTR;
```

_PRL_IO_SCREEN_UPDATE Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_CPULIMIT_DATA
Structure

Syntax

```
struct _PRL_CPULIMIT_DATA {
  PRL_UINT32 value;
  PRL_CPULIMIT_TYPE type;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_CVSRC_OPEN_MODE
Structure

Syntax

```
struct _PRL_CVSRC_OPEN_MODE {
  PRL_UINT32 sessionId;
  PRL_UINT32 frameFormat;
  PRL_UINT32 frameWidth;
  PRL_UINT32 frameHeight;
  PRL_UINT32 frameBytesPerRow;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_GET_VM_CONFIG_PARAM_DATA Structure

Syntax

```
struct _PRL_GET_VM_CONFIG_PARAM_DATA {
  PRL_VM_TYPE nVmType;
  PRL_CONST_STR sConfigSample;
  PRL_UINT32 nOsVersion;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT Structure

Syntax

```
struct _PRL_IO_CHANNEL_STAT {
  struct _PRL_CPU_FEATURES_MASK {
    struct _PRL_CPU_FEATURES {
      struct _PRL_HOST_OS_INFO {
        struct _PRL_MEMGUARANTEE_DATA {
          PRL_UINT32 value;
          PRL_MEMGUARANTEE_TYPE type;
        };
      };
    };
  };
};
```

File

PrlIOStructs.h

Members

_PRL_IO_CHANNEL_STAT Nested Types

Name	Description
PRL_IO_CHANNEL_STAT_PTR (p. 906)	

_PRL_IO_CHANNEL_STAT Structures

	Name	Description
	_PRL_CPU_FEATURES_MASK (p. 906)	

_PRL_IO_CHANNEL_STAT Nested Types

Name	Description
PRL_IO_CHANNEL_STAT_PTR (p. 906)	

_PRL_IO_CHANNEL_STAT Structures

	Name	Description
	_PRL_CPU_FEATURES_MASK (p. 906)	

_PRL_IO_CHANNEL_STAT Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::PRL_IO_CHANNEL_STAT_PTR
Nested Type

Syntax

```
typedef PRL_IO_CHANNEL_STAT* PRL_IO_CHANNEL_STAT_PTR;
```

_PRL_IO_CHANNEL_STAT Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK
Structure

Syntax

```
struct _PRL_CPU_FEATURES_MASK {
  struct _PRL_CPU_FEATURES {
    struct _PRL_HOST_OS_INFO {
      struct _PRL_MEMGUARANTEE_DATA {
        PRL_UINT32 value;
        PRL_MEMGUARANTEE_TYPE type;
      };
    };
  };
};
```

File

PrlIOStructs.h

Members

_PRL_CPU_FEATURES_MASK Nested Types

Name	Description
PRL_CONST_CPU_FEATURES_MASK_PTR (p. 907)	
PRL_CPU_FEATURES_MASK_PTR (p. 907)	

_PRL_CPU_FEATURES_MASK Structures

	Name	Description
	_PRL_CPU_FEATURES (p. 907)	

_PRL_CPU_FEATURES_MASK Nested Types

Name	Description
PRL_CONST_CPU_FEATURES_MASK_PTR (p. 907)	
PRL_CPU_FEATURES_MASK_PTR (p. 907)	

_PRL_CPU_FEATURES_MASK Structures

	Name	Description
	_PRL_CPU_FEATURES (p. 907)	

_PRL_CPU_FEATURES_MASK Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::PRL_CONST_CPU_FEATURES_MASK_PTR
Nested Type

Syntax

```
typedef const PRL_CPU_FEATURES_MASK* PRL_CONST_CPU_FEATURES_MASK_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::PRL_CPU_FEATURES_MASK_PTR
Nested Type

Syntax

```
typedef PRL_CPU_FEATURES_MASK* PRL_CPU_FEATURES_MASK_PTR;
```

_PRL_CPU_FEATURES_MASK Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES
Structure

Syntax

```
struct _PRL_CPU_FEATURES {
  struct _PRL_HOST_OS_INFO {
    struct _PRL_MEMGUARANTEE_DATA {
      PRL_UINT32 value;
      PRL_MEMGUARANTEE_TYPE type;
    };
  };
```

```
};
```

File

PrlIOStructs.h

Members

_PRL_CPU_FEATURES Nested Types

Name	Description
PRL_CPU_FEATURES_PTR (p. 908)	

_PRL_CPU_FEATURES Structures

	Name	Description
	_PRL_HOST_OS_INFO (p. 908)	

_PRL_CPU_FEATURES Nested Types

Name	Description
PRL_CPU_FEATURES_PTR (p. 908)	

_PRL_CPU_FEATURES Structures

	Name	Description
	_PRL_HOST_OS_INFO (p. 908)	

_PRL_CPU_FEATURES Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::PRL_CPU_FEATURES_PTR
Nested Type

Syntax

```
typedef PRL_CPU_FEATURES* PRL_CPU_FEATURES_PTR;
```

_PRL_CPU_FEATURES Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO
Structure

Syntax

```
struct _PRL_HOST_OS_INFO {
  struct _PRL_MEMGUARANTEE_DATA {
    PRL_UINT32 value;
    PRL_MEMGUARANTEE_TYPE type;
  };
};
```

File

PrlIOStructs.h

Members

_PRL_HOST_OS_INFO Nested Types

Name	Description
PRL_CONST_MEMGUARANTEE_DATA_PTR (p. 909)	
PRL_HOST_OS_INFO_PTR (p. 909)	
PRL_MEMGUARANTEE_DATA (p. 910)	
PRL_MEMGUARANTEE_DATA_PTR (p. 910)	

_PRL_HOST_OS_INFO Structures

	Name	Description
	_PRL_MEMGUARANTEE_DATA (p. 910)	

_PRL_HOST_OS_INFO Nested Types

Name	Description
PRL_CONST_MEMGUARANTEE_DATA_PTR (p. 909)	
PRL_HOST_OS_INFO_PTR (p. 909)	
PRL_MEMGUARANTEE_DATA (p. 910)	
PRL_MEMGUARANTEE_DATA_PTR (p. 910)	

_PRL_HOST_OS_INFO Structures

	Name	Description
	_PRL_MEMGUARANTEE_DATA (p. 910)	

_PRL_HOST_OS_INFO Nested Types

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_CONST_MEMGUARANTEE_DATA_PTR
Nested Type

Syntax

```
typedef const PRL_MEMGUARANTEE_DATA* PRL_CONST_MEMGUARANTEE_DATA_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_HOST_OS_INFO_PTR
Nested Type

Syntax

```
typedef PRL_HOST_OS_INFO* PRL_HOST_OS_INFO_PTR;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_MEMGUARANTEE_DATA
Nested Type

Syntax

```
typedef struct _PRL_MEMGUARANTEE_DATA {
  PRL_UINT32 value;
  PRL_MEMGUARANTEE_TYPE type;
} PRL_MEMGUARANTEE_DATA;
```

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_MEMGUARANTEE_DATA_PTR
Nested Type

Syntax

```
typedef PRL_MEMGUARANTEE_DATA* PRL_MEMGUARANTEE_DATA_PTR;
```

_PRL_HOST_OS_INFO Structures

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::_PRL_MEMGUARANTEE_DATA
Structure

Syntax

```
struct _PRL_MEMGUARANTEE_DATA {
  PRL_UINT32 value;
  PRL_MEMGUARANTEE_TYPE type;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_SSO_CLIENT_VM_PACKET
Structure

Syntax

```
struct _PRL_IO_SSO_CLIENT_VM_PACKET {
  PRL_UINT8 credentialsRequestUUID[16];
  PRL_UINT32 reserved[12];
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IOLIMIT_DATA
Structure

Syntax

```
struct _PRL_IOLIMIT_DATA {
  PRL_UINT32 value;
  PRL_IOLIMIT_TYPE type;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_STAT_NET_TRAFFIC
Structure

Syntax

```
struct _PRL_STAT_NET_TRAFFIC {
  PRL_UINT64 incoming[PRL_TC_CLASS_MAX];
  PRL_UINT64 outgoing[PRL_TC_CLASS_MAX];
  PRL_UINT32 incoming_pkt[PRL_TC_CLASS_MAX];
  PRL_UINT32 outgoing_pkt[PRL_TC_CLASS_MAX];
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_CLIENT_INFO
Structure

Syntax

```
struct _PRL_PROXY_CLIENT_INFO {
  PRL_UINT32 structSize;
  PRL_CHAR clientOptions[64];
  PRL_UINT8 customerSupportUUID[16];
  PRL_CHAR managerURL[256];
};
```

File

PrlIOStructs.h

Members

structSize

< Should be filled by the caller with actual structure size

clientOptions

< Reserved

customerSupportUUID

< Reserved

managerURL

< Reserved

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_DATA_KEY
Structure

Syntax

```
struct _PRL_PROXY_DATA_KEY {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  struct _PRL_PROXY_DATA_KEY * next;
};
```

File

PrlIOStructs.h

Members

dataKey

< Data identifier

dataVersion

< Data version

next

< Pointer to the next element

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_GEO_INFO
Structure

Syntax

```
struct _PRL_PROXY_GEO_INFO {
  PRL_CHAR countryName[64];
  PRL_CHAR countryCode[3];
  PRL_CHAR region[3];
  PRL_CHAR city[64];
  PRL_CHAR postalCode[10];
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_HOST_INFO
Structure

Syntax

```
struct _PRL_PROXY_HOST_INFO {
```

```
    PRL_UINT8 hostUUID[16];
    PRL_CHAR brokerAddresses[16][64];
    PRL_INT32 brokerAddressesSz;
    PRL_PROXY_REGISTRATION_INFO info;
  };
```

File

PrlIOStructs.h

Members

hostUUID

< Proxy host uuid

brokerAddresses

< Broker addresses to which connection should be established

brokerAddressesSz

< Broker addresses size

info

< Host registration info

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_HOST_INFO_EXT
Structure

Syntax

```
  struct _PRL_PROXY_HOST_INFO_EXT {
    PRL_PROXY_HOST_INFO hostInfo;
    PRL_BOOL isOnline;
    PRL_PROXY_GEO_INFO_PTR geoInfo;
    struct _PRL_PROXY_HOST_INFO_EXT * next;
  };
```

File

PrlIOStructs.h

Members

isOnline

Sign which specifies whether appropriate host online

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_REGISTRATION_INFO
Structure

Syntax

```
  struct _PRL_PROXY_REGISTRATION_INFO {
    PRL_CHAR hostAddresses[16][64];
    PRL_INT32 hostAddressesSz;
    PRL_CHAR hostName[64];
    PRL_UINT8 appUUID[16];
    PRL_CHAR appVersion[32];
    PRL_CHAR hostOptions[33];
    PRL_CHAR hostOsName[63];
    PRL_CHAR isoTimestamp[32];
```

```
};
```

File

PrlIOStructs.h

Members

hostAddresses
< Host addresses which host listens to

hostAddressesSz
< Size of host addresses

hostName
< Short host name

appUUID
< Application uuid

appVersion
< Application version

hostOptions
< Using for host icon hash transport

hostOsName
< Reserved

isoTimestamp
< Reserved

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_USER_DATA
Structure

Syntax

```
struct _PRL_PROXY_USER_DATA {
  PRL_STR dataKey;
  PRL_INT32 dataVersion;
  PRL_VOID_PTR data;
  PRL_UINT32 dataSize;
  struct _PRL_PROXY_USER_DATA * next;
};
```

File

PrlIOStructs.h

Members

dataKey
< Data identifier

dataVersion
< Data version

data
< Pointer to the data buffer

dataSize
< Data size

next
< Pointer to the next element

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT Structure

Syntax

```
struct _PRL_IO_MOUSE_EVENT {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
  PRL_UINT32 buttons;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT_EX Structure

Syntax

```
struct _PRL_IO_MOUSE_EVENT_EX {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
  PRL_UINT32 buttons;
  PRL_INT32 w;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT_EX2 Structure

Syntax

```
struct _PRL_IO_MOUSE_EVENT_EX2 {
  PRL_INT32 x;
  PRL_INT32 y;
  PRL_INT32 z;
  PRL_UINT32 buttons;
  PRL_INT32 w;
  PRL_INT32 z120;
  PRL_INT32 w120;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_SCREEN_SURFACE Structure

Syntax

```
struct _PRL_IO_SCREEN_SURFACE {
  PRL_UINT32 windowId;
  PRL_UINT32 surfaceId;
  PRL_INT32 scrollX;
  PRL_INT32 scrollY;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_SCREEN_REGION Structure

Syntax

```
struct _PRL_IO_DISPLAY_SCREEN_REGION {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 display;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_SCREEN_BUFFER_REGION Structure

Syntax

```
struct _PRL_IO_SCREEN_BUFFER_REGION {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 scaleWidth;
  PRL_UINT32 scaleHeight;
  PRL_UINT32 display;
};
```

File

PrlIOStructs.h

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_SCREEN_REGION Structure

Syntax

```
struct _PRL_IO_SCREEN_REGION {
  PRL_UINT32 x;
  PRL_UINT32 y;
  PRL_UINT32 width;
  PRL_UINT32 height;
};
```

File

PrlIOStructs.h

_PRL_IO_SCREEN_SIZE Structure

Syntax

```
struct _PRL_IO_SCREEN_SIZE {
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 depth;
  PRL_INT32 lineBytes;
};
```

File

PrlIOStructs.h

_PRL_IO_STATE Enumeration

Syntax

```
enum _PRL_IO_STATE {
  IOS_DISABLED = 0,
  IOS_STARTED = 1,
  IOS_STOPPED = 2,
  IOS_UNKNOWN_VM_UUID = 3,
  IOS_CONNECTION_TIMEOUT = 4,
  IOS_AUTH_FAILED = 5
};
```

File

PrlEnums.h

Members

IOS_DISABLED

< Desktop has not been enabled

IOS_STARTED

< Successfully started.(local and remote)

IOS_STOPPED

< Is stopped.

IOS_UNKNOWN_VM_UUID

< Unknown uuid.

IOS_CONNECTION_TIMEOUT
< Connection timeout.

IOS_AUTH_FAILED
< Authentication failed.

PRL_IO_CLIENT_FLAGS Enumeration

Flags which determines peer client while login to dispatcher and while attach to vm.

Syntax

```
typedef enum _PRL_IO_CLIENT_FLAGS {
  PCF_ORIGINAL_CLIENT = 1<<(PACF_MAX+1),
  PCF_LIGHTWEIGHT_CLIENT = 1<<(PACF_MAX+2),
  PCF_TRUSTED_CHANNEL = 1<<(PACF_MAX+3)
} PRL_IO_CLIENT_FLAGS;
```

File

PrlCommandsFlags.h

Members

PCF_ORIGINAL_CLIENT
Client with all features

PCF_LIGHTWEIGHT_CLIENT

- Lightweight client (iPhone/iPad) which

does not subscribe on heavy packages

PCF_TRUSTED_CHANNEL

- Login over trusted channel

(no need in username/password)

PRL_IO_DISCONNECT_REASON Enumeration

Specifies the reason of closing IO connection

Syntax

```
typedef enum _PRL_IO_DISCONNECT_REASON {
  PDR_UNSPECIFIED = 0,
  PDR_RECONNECT = 1,
  PDR_HOST_USER_LOGGED_IN = 2,
  PDR_HOST_USER_INPUT_DETECTED = 3,
  PDR_ANOTHER_CLIENT_CONNECTED = 4,
  PDR_SESSION_BECOME_INACTIVE = 5
} PRL_IO_DISCONNECT_REASON;
```

File

PrlEnums.h

PRL_IO_DISPLAY_SCREEN_SIZE Structure

Syntax

```
typedef struct _PRL_IO_DISPLAY_SCREEN_SIZE {
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 depth;
  PRL_INT32 lineBytes;
  PRL_UINT32 display;
  PRL_INT32 displayX;
  PRL_INT32 displayY;
} PRL_IO_DISPLAY_SCREEN_SIZE;
```

File

PrlIOStructs.h

PRL_IO_SCREEN_SIZE Structure

Syntax

```
typedef struct _PRL_IO_SCREEN_SIZE {
  PRL_UINT32 width;
  PRL_UINT32 height;
  PRL_UINT32 depth;
  PRL_INT32 lineBytes;
} PRL_IO_SCREEN_SIZE;
```

File

PrlIOStructs.h

PRL_IO_STATE Enumeration

Syntax

```
typedef enum _PRL_IO_STATE {
  IOS_DISABLED = 0,
  IOS_STARTED = 1,
  IOS_STOPPED = 2,
  IOS_UNKNOWN_VM_UUID = 3,
  IOS_CONNECTION_TIMEOUT = 4,
  IOS_AUTH_FAILED = 5
} PRL_IO_STATE;
```

File

PrlEnums.h

Members

IOS_DISABLED

< Desktop has not been enabled

IOS_STARTED

< Successfully started.(local and remote)

IOS_STOPPED

< Is stopped.

IOS_UNKNOWN_VM_UUID

< Unknown uuid.

IOS_CONNECTION_TIMEOUT

< Connection timeout.

IOS_AUTH_FAILED

< Authentication failed.

PRL_IOLIMIT_TYPE Enumeration

Syntax

```
typedef enum _PRL_IOLIMIT_TYPE {
  PRL_IOLIMIT_BS = 1
} PRL_IOLIMIT_TYPE;
```

File

PrlEnums.h

PRL_VM Structs

Enumerations

	Name	Description
	_PRL_VM_ACCELERATION_LEVEL (p. 923)	Acceleration level
	_PRL_VM_ACCESS_FOR_OTHERS (p. 923)	Specifies virtual machine access rights.
	_PRL_VM_ACTION (p. 924)	
	_PRL_VM_ACTION_ON_STOP (p. 924)	Action when VM gets stopped
	_PRL_VM_ACTION_ON_WINDOW_CLOSE (p. 924)	Action on VM when VM window gets closed
	_PRL_VM_AUTOSTART_OPTION (p. 925)	VM autostart options
	_PRL_VM_AUTOSTOP_OPTION (p. 925)	VM autostop options
	_PRL_VM_BACKUP_FLAGS (p. 925)	VM backup functionality flags set
	_PRL_VM_BACKUP_TYPE (p. 926)	Hint about the type of VE inside a backup archive
	_PRL_VM_BOOT_ORDER (p. 927)	VM boot order
	_PRL_VM_COLOR (p. 927)	VM colors
	_PRL_VM_COMMIT_FLAGS (p. 927)	PrlVm_CommitEx (p. 455) flags
	_PRL_VM_CONFIG_SECTIONS (p. 928)	VM configuration sections
	_PRL_VM_CPULIMIT_TYPE (p. 929)	

	_PRL_VM_DEV_EMULATION_TYPE (p. 929)	VM device emulation type
	_PRL_VM_EXEC_APP_FILE_DESCRIPTOR (p. 929)	App execution result extraction moderators: PFD_STDOUT : request stdout of executed in guest OS app PFD_STDERR : request stderr of executed in guest OS app
	_PRL_VM_EXEC_RUN_PROGRAM_MODES (p. 930)	Run app in guest modes: PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY : just run program in guest OS and do not wait result PRPM_RUN_PROGRAM_IN_SHELL : run program via shell and in shell environment PRPM_RUN_PROGRAM_ENTER : run shell in guest
	_PRL_VM_LOCATION (p. 930)	VM files location set PVL_UNKNOWN : VM files location unknown (mostly for invalid VMs) PVL_LOCAL_FS : VM files placed on local file system PVL_REMOTE_FS : VM files placed on remote file system PVL_USB_DRIVE : VM files placed on external USB drive PVL_FIREWIRE_DRIVE : VM files placed on external FireWire drive
	_PRL_VM_MIGRATION_FLAGS (p. 931)	VM migration functionality flags set
	_PRL_VM_MOUSE_CURSOR_STATE (p. 932)	
	_PRL_VM_MOUSE_STATE (p. 932)	
	_PRL_VM_NET_ADAPTER_TYPE (p. 932)	
	_PRL_VM_NET_ADAPTER_WIFI_RULE (p. 933)	
	_PRL_VM_NORMAL_START_FLAGS (p. 933)	VM normal-mode start flags: PSM_VM_START_IGNORE_HA_CLUSTER : do not register VM on HA cluster in any case (for server mode only) PNSF_CT_SKIP_ARPDETECT : Ct specific parameter
	_PRL_VM_PRIORITY (p. 933)	Background/foreground VM processes priority
	_PRL_VM_REMOTE_DISPLAY_MODE (p. 934)	Virtuozzo remote desktop modes
	_PRL_VM_SMARTGUARD_SCHEMA (p. 934)	SmartGuard schema
	_PRL_VM_START_LOGIN_MODE (p. 934)	VM start login mode
	_PRL_VM_START_MODE (p. 935)	VM start mode: PSM_VM_START : start VM PSM_VM_SAFE_START : start VM in the safe mode PSM_VM_START_FOR_COMPACT : start VM only for compacting disk operation
	_PRL_VM_STATISTICS_FLAGS (p. 935)	PrlVm_GetStatisticsEx (p. 470) flags
	_PRL_VM_STOP_FLAGS (p. 935)	Virtual machine shutdown options.
	_PRL_VM_STOP_MODE (p. 936)	Vm stop modes
	_PRL_VM_STORE_VALUE_FLAGS (p. 936)	Store value by specified keys on server options.
	_PRL_VM_TOOLS_STATE (p. 936)	Virtual machine tools states enumeration
	_PRL_VM_TYPE (p. 937)	
	_PRL_VM_TYPE_FLAGS (p. 937)	
	_PRL_VM_WINDOW_MODE (p. 937)	VM window mode
	PRL_VM_ACCELERATION_LEVEL (p. 938)	Acceleration level
	PRL_VM_ACCESS_FOR_OTHERS (p. 938)	Specifies virtual machine access rights.
	PRL_VM_ACTION (p. 938)	
	PRL_VM_ACTION_ON_STOP (p. 939)	Action when VM gets stopped

	PRL_VM_ACTION_ON_WINDOW_CLOSE (p. 939)	Action on VM when VM window gets closed
	PRL_VM_AUTOSTART_OPTION (p. 939)	VM autostart options
	PRL_VM_AUTOSTOP_OPTION (p. 940)	VM autostop options
	PRL_VM_BACKUP_FLAGS (p. 940)	VM backup functionality flags set
	PRL_VM_BACKUP_TYPE (p. 941)	Hint about the type of VE inside a backup archive
	PRL_VM_BOOT_ORDER (p. 941)	VM boot order
	PRL_VM_COLOR (p. 942)	VM colors
	PRL_VM_COMMIT_FLAGS (p. 942)	PrlVm_CommitEx (p. 455) flags
	PRL_VM_CONFIG_SECTIONS (p. 943)	VM configuration sections
	PRL_VM_CPULIMIT_TYPE (p. 943)	
	PRL_VM_DEV_EMULATION_TYPE (p. 944)	VM device emulation type
	PRL_VM_EXEC_APP_FILE_DESCRIPTOR (p. 944)	App execution result extraction moderators: PFD_STDOUT : request stdout of executed in guest OS app PFD_STDERR : request stderr of executed in guest OS app
	PRL_VM_EXEC_RUN_PROGRAM_MODES (p. 945)	Run app in guest modes: PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY : just run program in guest OS and do not wait result PRPM_RUN_PROGRAM_IN_SHELL : run program via shell and in shell environment PRPM_RUN_PROGRAM_ENTER : run shell in guest
	PRL_VM_LOCATION (p. 945)	VM files location set PVL_UNKNOWN : VM files location unknown (mostly for invalid VMs) PVL_LOCAL_FS : VM files placed on local file system PVL_REMOTE_FS : VM files placed on remote file system PVL_USB_DRIVE : VM files placed on external USB drive PVL_FIREWIRE_DRIVE : VM files placed on external FireWire drive
	PRL_VM_MIGRATION_FLAGS (p. 945)	VM migration functionality flags set
	PRL_VM_MOUSE_CURSOR_STATE (p. 946)	
	PRL_VM_MOUSE_STATE (p. 947)	
	PRL_VM_NET_ADAPTER_TYPE (p. 947)	
	PRL_VM_NET_ADAPTER_WIFI_RULE (p. 947)	
	PRL_VM_NORMAL_START_FLAGS (p. 947)	VM normal-mode start flags: PSM_VM_START_IGNORE_HA_CLUSTER : do not register VM on HA cluster in any case (for server mode only) PNSF_CT_SKIP_ARPDETECT : Ct specific parameter
	PRL_VM_ON_CRASH_ACTION (p. 948)	Specifies an on-crash actions
	PRL_VM_ON_CRASH_OPTION (p. 948)	Specifies an on-crash options
	PRL_VM_PRIORITY (p. 948)	Background/foreground VM processes priority
	PRL_VM_REMOTE_DISPLAY_MODE (p. 949)	Virtuozzo remote desktop modes
	PRL_VM_SMARTGUARD_SCHEMA (p. 949)	SmartGuard schema
	PRL_VM_START_LOGIN_MODE (p. 949)	VM start login mode
	PRL_VM_START_MODE (p. 950)	VM start mode: PSM_VM_START : start VM PSM_VM_SAFE_START : start VM in the safe mode PSM_VM_START_FOR_COMPACT : start VM only for compacting disk operation
	PRL_VM_STATISTICS_FLAGS (p. 950)	PrlVm_GetStatisticsEx (p. 470) flags
	PRL_VM_STOP_FLAGS (p. 950)	Virtual machine shutdown options.

	PRL_VM_STOP_MODE (p. 951)	Vm stop modes
	PRL_VM_STORE_VALUE_FLAGS (p. 951)	Store value by specified keys on server options.
	PRL_VM_TOOLS_STATE (p. 952)	Virtual machine tools states enumeration
	PRL_VM_TYPE (p. 952)	
	PRL_VM_TYPE_FLAGS (p. 952)	
	PRL_VM_WINDOW_MODE (p. 952)	VM window mode

_PRL_VM_ACCELERATION_LEVEL Enumeration

Acceleration level

Syntax

```
enum _PRL_VM_ACCELERATION_LEVEL {
  PVA_ACCELERATION_DISABLED = 0,
  PVA_ACCELERATION_NORMAL = 1,
  PVA_ACCELERATION_HIGH = 2
};
```

File

PrlEnums.h

_PRL_VM_ACCESS_FOR_OTHERS Enumeration

Specifies virtual machine access rights.

Syntax

```
enum _PRL_VM_ACCESS_FOR_OTHERS {
  PAO_VM_NOT_SHARED = 0,
  PAO_VM_SHARED_ON_VIEW,
  PAO_VM_SHARED_ON_VIEW_AND_RUN,
  PAO_VM_SHARED_ON_FULL_ACCESS
};
```

File

PrlEnums.h

Type Parameters

PAO_VM_NOT_SHARED

Only the owner of the virtual machine has access to it. The machine will be hidden from all other users.

PAO_VM_SHARED_ON_VIEW

Other users can see the machine but they cannot run it. Other access rights can be determined using methods from the PHT_ACCESS_RIGHTS (p. 9) object.

PAO_VM_SHARED_ON_VIEW_AND_RUN

Other users can see the machine and they can also run it. Other access rights can be determined using methods from the PHT_ACCESS_RIGHTS (p. 9) object.

PAO_VM_SHARED_ON_FULL_ACCESS

Other users have full access to the virtual machine.

_PRL_VM_ACTION Enumeration

Syntax

```
enum _PRL_VM_ACTION {
  PVA_PRESTART = 0,
  PVA_POSTSTART = 1,
  PVA_PRESTOP = 2,
  PVA_POSTSTOP = 3
};
```

File

PrlEnums.h

_PRL_VM_ACTION_ON_STOP Enumeration

Action when VM gets stopped

Syntax

```
enum _PRL_VM_ACTION_ON_STOP {
  PAS_KEEP_VM_WINDOW_OPEN = 0,
  PAS_CLOSE_VM_WINDOW = 1,
  PAS_QUIT_APPLICATION = 2
};
```

File

PrlEnums.h

_PRL_VM_ACTION_ON_WINDOW_CLOSE Enumeration

Action on VM when VM window gets closed

Syntax

```
enum _PRL_VM_ACTION_ON_WINDOW_CLOSE {
  PWC_VM_UNKNOWN_ACTION = 0xffff,
  PWC_VM_STOP = 0,
  PWC_VM_SUSPEND = 1,
  PWC_VM_ASK_USER = 2,
  PWC_VM_PAUSE = 3,
  PWC_VM_SHUTDOWN = 4,
  PWC_VM_DO_NOTHING = 5
};
```

File

PrlEnums.h

_PRL_VM_AUTOSTART_OPTION Enumeration

VM autostart options

Syntax

```
enum _PRL_VM_AUTOSTART_OPTION {
  PAO_VM_START_MANUAL = 0,
  PAO_VM_START_ON_LOAD = 1,
  PAO_VM_START_ON_RELOAD = 2,
  PAO_VM_START_ON_GUI_APP_STARTUP = 3,
  PAO_VM_START_ON_GUI_VM_WINDOW_OPEN = 4,
  PAO_VM_START_ON_GUI_WINDOW_LOAD = PAO_VM_START_ON_GUI_APP_STARTUP
};
```

File

PrlEnums.h

Members

PAO_VM_START_ON_GUI_WINDOW_LOAD
OBSOLETE CONSTANTS

_PRL_VM_AUTOSTOP_OPTION Enumeration

VM autostop options

Syntax

```
enum _PRL_VM_AUTOSTOP_OPTION {
  PAO_VM_STOP = 0,
  PAO_VM_SUSPEND = 1,
  PAO_VM_SHUTDOWN = 2
};
```

File

PrlEnums.h

_PRL_VM_BACKUP_FLAGS Enumeration

VM backup functionality flags set

Syntax

```
enum _PRL_VM_BACKUP_FLAGS {
  PBSL_LOW_SECURITY = PSL_LOW_SECURITY,
  PBSL_NORMAL_SECURITY = PACF_NORMAL_SECURITY,
  PBSL_HIGH_SECURITY = PACF_HIGH_SECURITY,
  PBT_FULL = 1<<(PACF_MAX+1),
  PBT_INCREMENTAL = 1<<(PACF_MAX+2),
  PBT_DIFFERENTIAL = 1<<(PACF_MAX+3),
  PBT_RESTORE_TO_COPY = 1<<(PACF_MAX+4),
  PBT_IGNORE_NOT_EXISTS = 1<<(PACF_MAX+5),
  PBT_UNCOMPRESSED = 1<<(PACF_MAX+6),
```

```
    PBT_VM = 1<<(PACF_MAX+7),
    PBT_CT = 1<<(PACF_MAX+8),
    PBT_BACKUP_ID = 1<<(PACF_MAX+9),
    PBT_CHAIN = 1<<(PACF_MAX+10),
    PBT_KEEP_CHAIN = 1<<(PACF_MAX+11),
    PBT_DIRECT_DATA_CONNECTION = 1<<(PACF_MAX+12)
  };
```

File

PrlCommandsFlags.h

Type Parameters

PBT_FULL
: full backup

PBT_INCREMENTAL
: incremental backup

PBT_DIFFERENTAL
: differental backup

PBT_RESTORE_TO_COPY
: change Vm name/uuid/MAC address on restore

PBT_UNCOMPRESSED
: uncompressed backup

PBT_VM
: for virtual mashines

PBT_CT
: for containers

PBT_BACKUP_ID
: get information about a certain backup

PBT_CHAIN
: get information about the whole backup chain for a certain backup

PBT_KEEP_CHAIN
: preserve the rest of the backup chain on backup removal

PBT_DIRECT_DATA_CONNECTION
: disables tunneling for backup (a laissez-fair data transfer)

_PRL_VM_BACKUP_TYPE Enumeration

Hint about the type of VE inside a backup archive

Syntax

```
enum _PRL_VM_BACKUP_TYPE {
  PVBT_VM = 0,
  PVBT_CT_VZFS = 1,
  PVBT_CT_PLOOP = 2,
  PVBT_CT_VZWIN = 3
```

```
};
```

File

PrlEnums.h

_PRL_VM_BOOT_ORDER Enumeration

VM boot order

Syntax

```
enum _PRL_VM_BOOT_ORDER {
  PBO_HDD_CD_FLOPPY = 0,
  PBO_FLOPPY_HDD_CD = 1,
  PBO_CD_HDD_FLOPPY = 2
};
```

File

PrlEnums.h

_PRL_VM_COLOR Enumeration

VM colors

Syntax

```
enum _PRL_VM_COLOR {
  PVC_COLOR_RED = 0x00FA645A,
  PVC_COLOR_ORANGE = 0x00F6AA44,
  PVC_COLOR_YELLOW = 0x00EFDB47,
  PVC_COLOR_GREEN = 0x00B4D747,
  PVC_COLOR_BLUE = 0x005AA3FF,
  PVC_COLOR_PURPLE = 0x00C08ED8,
  PVC_COLOR_GREY = 0x00808080,
  PVC_NO_COLOR = 0x00000000
};
```

File

PrlEnums.h

_PRL_VM_COMMIT_FLAGS Enumeration

PrlVm_CommitEx (p. 455) flags

Syntax

```
enum _PRL_VM_COMMIT_FLAGS {
  PVCF_DESTROY_HDD_BUNDLE = 1<<(PACF_MAX+1),
  PVCF_WAIT_FOR_APPLY = 1<<(PACF_MAX+2),
  PEVF_WAIT_FOR_APPLY = PVCF_WAIT_FOR_APPLY,
  PVCF_DESTROY_HDD_BUNDLE_FORCE = 1<<(PACF_MAX+3),
  PVCF_DETACH_HDD_BUNDLE = 1<<(PACF_MAX+4),
  PVCF_RENAME_EXT_DISKS = 1<<(PACF_MAX+5)
```

```
};
```

File

PrlCommandsFlags.h

Members

PVCF_DESTROY_HDD_BUNDLE
specifies to destroy HDD bundle physically on hard disk remove operation

PVCF_WAIT_FOR_APPLY
wait for apply

PEVF_WAIT_FOR_APPLY
to provide compatibility

PVCF_DESTROY_HDD_BUNDLE_FORCE
destroy HDD bundle physically even if used in snapshots

_PRL_VM_CONFIG_SECTIONS Enumeration

VM configuration sections

Syntax

```
enum _PRL_VM_CONFIG_SECTIONS {
  PVC_VALIDATE_CHANGES_ONLY = 0,
  PVC_ALL = 1,
  PVC_GENERAL_PARAMETERS = 2,
  PVC_BOOT_OPTION = 3,
  PVC_REMOTE_DISPLAY = 4,
  PVC_SHARED_FOLDERS = 5,
  PVC_CPU = 6,
  PVC_MAIN_MEMORY = 7,
  PVC_VIDEO_MEMORY = 8,
  PVC_FLOPPY_DISK = 9,
  PVC_CD_DVD_ROM = 10,
  PVC_HARD_DISK = 11,
  PVC_NETWORK_ADAPTER = 12,
  PVC_SOUND = 13,
  PVC_SERIAL_PORT = 14,
  PVC_PARALLEL_PORT = 15,
  PVC_IDE_DEVICES = 16,
  PVC_SCSI_DEVICES = 17,
  PVC_GENERIC_PCI = 18,
  PVC_OFFLINE_MANAGEMENT_SETTINGS = 19,
  PVC_LICENSE_RESTRICTIONS = 20,
  PVC_SATA_DEVICES = 21,
  PVC_LAST_SECTION = PVC_SATA_DEVICES
};
```

File

PrlEnums.h

_PRL_VM_CPULIMIT_TYPE Enumeration

Syntax

```
enum _PRL_VM_CPULIMIT_TYPE {
  PRL_VM_CPULIMIT_FULL = 0,
  PRL_VM_CPULIMIT_GUEST = 1
};
```

File

PrlEnums.h

_PRL_VM_DEV_EMULATION_TYPE Enumeration

VM device emulation type

Syntax

```
enum _PRL_VM_DEV_EMULATION_TYPE {
  PDT_ANY_TYPE = 0xffff,
  PDT_USE_REAL_DEVICE = 0,
  PDT_USE_IMAGE_FILE = 1,
  PDT_USE_OUTPUT_FILE = 2,
  PDT_USE_OTHER = 3,
  PDT_USE_DIRECT_ASSIGN = 4,
  PDT_USE_FILE_SYSTEM = 5,
  PDT_USE_TCP = 6,
  PDT_USE_UDP = 7,
  PDT_USE_REAL_FLOPPY = PDT_USE_REAL_DEVICE,
  PDT_USE_REAL_HDD = PDT_USE_REAL_DEVICE,
  PDT_USE_REAL_CDDVD = PDT_USE_REAL_DEVICE,
  PDT_USE_REAL_SERIAL_PORT = PDT_USE_REAL_DEVICE,
  PDT_USE_SERIAL_PORT_SOCKET_MODE = PDT_USE_OTHER,
  PDT_USE_SERIAL_PORT_TCP_MODE = PDT_USE_TCP,
  PDT_USE_SERIAL_PORT_UDP_MODE = PDT_USE_UDP,
  PDT_USE_REAL_PARALLEL_PORT = PDT_USE_REAL_DEVICE,
  PDT_USE_PARALLEL_PORT_PRINTER_MODE = PDT_USE_OTHER,
  PDT_USE_PARALLEL_PORT_PRINT_TO_PDF_MODE = PDT_USE_IMAGE_FILE,
  PDT_USE_HOST_ONLY_NETWORK = PDT_USE_REAL_DEVICE,
  PDT_USE_SHARED_NETWORK = PDT_USE_IMAGE_FILE,
  PDT_USE_BRIDGE_ETHERNET = PDT_USE_OUTPUT_FILE,
  PDT_USE_AC97_SOUND = PDT_USE_REAL_DEVICE,
  PDT_USE_CREATIVE_SB16_SOUND = PDT_USE_IMAGE_FILE,
  PDT_USE_REAL_USB_CONTROLLER = PDT_USE_REAL_DEVICE,
  PDT_USE_BOOTCAMP = PDT_USE_OTHER
};
```

File

PrlEnums.h

_PRL_VM_EXEC_APP_FILE_DESCRIPTOR Enumeration

App execution result extraction moderators: PFD_STDOUT : request stdout of executed in guest OS app PFD_STDERR :

request stderr of executed in guest OS app

Syntax

```
enum _PRL_VM_EXEC_APP_FILE_DESCRIPTOR {
  PFD_STDOUT = 1<<(PACF_MAX+1),
  PFD_STDERR = 1<<(PACF_MAX+2),
  PFD_STDIN = 1<<(PACF_MAX+3),
  PFD_ALL = (PFD_STDOUT | PFD_STDERR | PFD_STDIN)
};
```

File

PrlCommandsFlags.h

_PRL_VM_EXEC_RUN_PROGRAM_MODES Enumeration

Run app in guest modes: PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY : just run program in guest OS and do not wait result PRPM_RUN_PROGRAM_IN_SHELL : run program via shell and in shell environment PRPM_RUN_PROGRAM_ENTER : run shell in guest

Syntax

```
enum _PRL_VM_EXEC_RUN_PROGRAM_MODES {
  PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY = 1<<(PACF_MAX+4),
  PRPM_RUN_PROGRAM_IN_SHELL = 1<<(PACF_MAX+5),
  PRPM_RUN_PROGRAM_ENTER = 1<<(PACF_MAX+6)
};
```

File

PrlCommandsFlags.h

_PRL_VM_LOCATION Enumeration

VM files location set PVL_UNKNOWN : VM files location unknown (mostly for invalid VMs) PVL_LOCAL_FS : VM files placed on local file system PVL_REMOTE_FS : VM files placed on remote file system PVL_USB_DRIVE : VM files placed on external USB drive PVL_FIREWIRE_DRIVE : VM files placed on external FireWire drive

Syntax

```
enum _PRL_VM_LOCATION {
  PVL_UNKNOWN = 0,
  PVL_LOCAL_FS = 1,
  PVL_REMOTE_FS = 2,
  PVL_USB_DRIVE = 3,
  PVL_FIREWIRE_DRIVE = 4
};
```

File

PrlEnums.h

_PRL_VM_MIGRATION_FLAGS Enumeration

VM migration functionality flags set

Syntax

```
enum _PRL_VM_MIGRATION_FLAGS {
  PVMSL_LOW_SECURITY = PSL_LOW_SECURITY,
  PVMSL_NORMAL_SECURITY = PACF_NORMAL_SECURITY,
  PVMSL_HIGH_SECURITY = PACF_HIGH_SECURITY,
  PVMT_COLD_MIGRATION = 1<<(PACF_MAX+1),
  PVMT_WARM_MIGRATION = 1<<(PACF_MAX+2),
  PVMT_HOT_MIGRATION = 1<<(PACF_MAX+3),
  PVM_DONT_RESUME_VM = 1<<(PACF_MAX+4),
  PVMT_DONT_RESUME_VM = PVM_DONT_RESUME_VM,
  PVMT_CLONE_MODE = 1<<(PACF_MAX+5),
  PVMT_SWITCH_TEMPLATE = 1<<(PACF_MAX+6),
  PVMT_CHANGE_SID = 1<<(PACF_MAX+7),
  PVMT_IGNORE_EXISTING_BUNDLE = 1<<(PACF_MAX+8),
  PVMT_REMOVE_SOURCE_BUNDLE = 1<<(PACF_MAX+9),
  PVMT_SEND_DISK_MAP = 1<<(PACF_MAX+10),
  PVMT_UNCOMPRESSED = 1<<(PACF_MAX+11),
  PVMT_DIRECT_DATA_CONNECTION = 1<<(PACF_MAX+12),
  PVMT_DONT_CREATE_DISK = 1<<(PACF_MAX+13)
};
```

File

PrlCommandsFlags.h

Type Parameters

PVMT_COLD_MIGRATION

VM migration with interruption of VM services (VM will be stopped on migration time).

PVMT_WARM_MIGRATION

VM services will have quite long downtime during migration but won't be interrupted.

PVMT_HOT_MIGRATION

VM services will have short downtime during migration without any interruption.

PVMT_DONT_RESUME_VM

do not start running VM after migration

PVMT_CLONE_MODE

preserves source VM

PVMT_SWITCH_TEMPLATE

applable just in clone mode - lets to clone remotely VM to template

PVMT_CHANGE_SID

for Windows VMs/containers lets to change SID after remote clone

PVMT_IGNORE_EXISTING_BUNDLE

do not fail if Vm bundle exist. For clone Vm on mounted iSCSI storage

PVMT_UNCOMPRESSED

do not compress data during migration

PVMT_DIRECT_DATA_CONNECTION
do not use tunnel for migration

PVMT_DONT_CREATE_DISK
don't create disk file on target host

_PRL_VM_MOUSE_CURSOR_STATE Enumeration

Syntax

```
enum _PRL_VM_MOUSE_CURSOR_STATE {
  PTS_NO_CURSOR = 0,
  PTS_SLIDING_CURSOR = 1,
  PTS_MOVING_CURSOR = 2
};
```

File

PrlEnums.h

_PRL_VM_MOUSE_STATE Enumeration

Syntax

```
enum _PRL_VM_MOUSE_STATE {
  PTS_RELATIVE_MOUSE = 0,
  PTS_SLIDING_MOUSE = 1,
  PTS_ABSOLUTE_MOUSE = 2
};
```

File

PrlEnums.h

_PRL_VM_NET_ADAPTER_TYPE Enumeration

Syntax

```
enum _PRL_VM_NET_ADAPTER_TYPE {
  PNT_UNDEFINED = 0,
  PNT_RTL = 1,
  PNT_E1000 = 2,
  PNT_VIRTIO = 3,
  PNT_E1000E = 4
};
```

File

PrlEnums.h

_PRL_VM_NET_ADAPTER_WIFI_RULE Enumeration

Syntax

```
enum _PRL_VM_NET_ADAPTER_WIFI_RULE {
  PNWR_GENERATED_MAC = 0,
  PNWR_HOST_MAC = 1,
  PNWR_AUTO = 2
};
```

File

PrlEnums.h

_PRL_VM_NORMAL_START_FLAGS Enumeration

VM normal-mode start flags: PSM_VM_START_IGNORE_HA_CLUSTER : do not register VM on HA cluster in any case (for server mode only) PNSF_CT_SKIP_ARPDETECT : Ct specific parameter

Syntax

```
enum _PRL_VM_NORMAL_START_FLAGS {
  PNSF_VM_START_WAIT = 1<<(PACF_MAX+2),
  PNSF_CT_SKIP_ARPDETECT = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

Members

PNSF_VM_START_WAIT
PNSF_VM_START_IGNORE_HA_CLUSTER = 1<<(PACF_MAX+1), This flag is unused

_PRL_VM_PRIORITY Enumeration

Background/foreground VM processes priority

Syntax

```
enum _PRL_VM_PRIORITY {
  PVR_PRIORITY_LOW = 0,
  PVR_PRIORITY_NORMAL = 1,
  PVR_PRIORITY_HIGH = 2
};
```

File

PrlEnums.h

_PRL_VM_REMOTE_DISPLAY_MODE Enumeration

Virtuozzo remote desktop modes

Syntax

```
enum _PRL_VM_REMOTE_DISPLAY_MODE {
  PRD_DISABLED = 0,
  PRD_AUTO,
  PRD_MANUAL
};
```

File

PrlEnums.h

_PRL_VM_SMARTGUARD_SCHEMA Enumeration

SmartGuard schema

Syntax

```
enum _PRL_VM_SMARTGUARD_SCHEMA {
  PVSS_OPTIMIZED_FOR_TIME_MACHINE = 1,
  PVSS_CUSTOM = 2
};
```

File

PrlEnums.h

Members

PVSS_OPTIMIZED_FOR_TIME_MACHINE

Automatic schema for time machine

PVSS_CUSTOM

User defined schema

_PRL_VM_START_LOGIN_MODE Enumeration

VM start login mode

Syntax

```
enum _PRL_VM_START_LOGIN_MODE {
  PLM_START_ACCOUNT = 0,
  PLM_ROOT_ACCOUNT = 1,
  PLM_USER_ACCOUNT = 2,
  PLM_START_ACOUNT = PLM_START_ACCOUNT,
  PLM_ROOT_ACOUNT = PLM_ROOT_ACCOUNT,
  PLM_USER_ACOUNT = PLM_USER_ACCOUNT
};
```

File

PrlEnums.h

Members

PLM_START_ACOUNT
Obsolete values for compatibility

_PRL_VM_START_MODE Enumeration

VM start mode: PSM_VM_START : start VM PSM_VM_SAFE_START : start VM in the safe mode PSM_VM_START_FOR_COMPACT : start VM only for compacting disk operation

Syntax

```
enum _PRL_VM_START_MODE {
  PSM_VM_START = 1<<(PACF_MAX+1),
  PSM_VM_SAFE_START = 1<<(PACF_MAX+2),
  PSM_VM_START_FOR_COMPACT = 1<<(PACF_MAX+3),
  PSM_VM_START_LAST_ITEM = PSM_VM_START_FOR_COMPACT
};
```

File

PrlCommandsFlags.h

Members

PSM_VM_START_LAST_ITEM
PSM_VM_START_LAST_ITEM has to be equal the last mode

_PRL_VM_STATISTICS_FLAGS Enumeration

PrlVm_GetStatisticsEx (p. 470) flags

Syntax

```
enum _PRL_VM_STATISTICS_FLAGS {
  PVMSF_HOST_DISK_SPACE_USAGE_ONLY = 1<<(PACF_MAX+1),
  PVMSF_GUEST_OS_INFORMATION_ONLY = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

_PRL_VM_STOP_FLAGS Enumeration

Virtual machine shutdown options.

Syntax

```
enum _PRL_VM_STOP_FLAGS {
  PSF_FORCE = 1<<(PACF_MAX+1),
```

```
  PSF_NOFORCE = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

Remarks

- PSF_FORCE : forcibly stop the virtual machine if shutdown timed out.
- PSF_NOFORCE : do nothing if the virtual machine shutdown timed out.

_PRL_VM_STOP_MODE Enumeration

Vm stop modes

Syntax

```
enum _PRL_VM_STOP_MODE {
  PSM_KILL = 0,
  PSM_SHUTDOWN = 1,
  PSM_ACPI = 2,
  PSM_LAST = PSM_ACPI
};
```

File

PrlEnums.h

Members

PSM_LAST

Should equals to the last item

_PRL_VM_STORE_VALUE_FLAGS Enumeration

Store value by specified keys on server options.

Syntax

```
enum _PRL_VM_STORE_VALUE_FLAGS {
  PSV_CALC_BOOT_CAMP_SIZE = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Remarks

- PSV_CALC_BOOT_CAMP : calculate VM size with Boot Camp disks

_PRL_VM_TOOLS_STATE Enumeration

Virtual machine tools states enumeration

Syntax

```
enum _PRL_VM_TOOLS_STATE {
  PTS_UNKNOWN = 0,
  PTS_POSSIBLY_INSTALLED = PTS_UNKNOWN,
  PTS_NOT_INSTALLED = 1,
  PTS_INSTALLED = 2,
  PTS_OUTDATED = 3
};
```

File

PrlEnums.h

_PRL_VM_TYPE Enumeration

Syntax

```
enum _PRL_VM_TYPE {
  PVT_VM = 0,
  PVT_CT = 1
};
```

File

PrlEnums.h

_PRL_VM_TYPE_FLAGS Enumeration

Syntax

```
enum _PRL_VM_TYPE_FLAGS {
  PVTF_VM = 1<<(PACF_MAX+1),
  PVTF_CT = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

_PRL_VM_WINDOW_MODE Enumeration

VM window mode

Syntax

```
enum _PRL_VM_WINDOW_MODE {
  PWM_DEFAULT_WINDOW_MODE = 0,
  PWM_WINDOWED_WINDOW_MODE = 1,
  PWM_FULL_SCREEN_WINDOW_MODE = 2
};
```

File

PrlEnums.h

PRL_VM_ACCELERATION_LEVEL Enumeration

Acceleration level

Syntax

```
typedef enum _PRL_VM_ACCELERATION_LEVEL {
  PVA_ACCELERATION_DISABLED = 0,
  PVA_ACCELERATION_NORMAL = 1,
  PVA_ACCELERATION_HIGH = 2
} PRL_VM_ACCELERATION_LEVEL;
```

File

PrlEnums.h

PRL_VM_ACCESS_FOR_OTHERS Enumeration

Specifies virtual machine access rights.

Syntax

```
typedef enum _PRL_VM_ACCESS_FOR_OTHERS {
  PAO_VM_NOT_SHARED = 0,
  PAO_VM_SHARED_ON_VIEW,
  PAO_VM_SHARED_ON_VIEW_AND_RUN,
  PAO_VM_SHARED_ON_FULL_ACCESS
} PRL_VM_ACCESS_FOR_OTHERS;
```

File

PrlEnums.h

Type Parameters

PAO_VM_NOT_SHARED

Only the owner of the virtual machine has access to it. The machine will be hidden from all other users.

PAO_VM_SHARED_ON_VIEW

Other users can see the machine but they cannot run it. Other access rights can be determined using methods from the PHT_ACCESS_RIGHTS (p. 9) object.

PAO_VM_SHARED_ON_VIEW_AND_RUN

Other users can see the machine and they can also run it. Other access rights can be determined using methods from the PHT_ACCESS_RIGHTS (p. 9) object.

PAO_VM_SHARED_ON_FULL_ACCESS

Other users have full access to the virtual machine.

PRL_VM_ACTION Enumeration

Syntax

```
typedef enum _PRL_VM_ACTION {
```

```
  PVA_PRESTART = 0,
  PVA_POSTSTART = 1,
  PVA_PRESTOP = 2,
  PVA_POSTSTOP = 3
} PRL_VM_ACTION;
```

File

PrlEnums.h

PRL_VM_ACTION_ON_STOP Enumeration

Action when VM gets stopped

Syntax

```
typedef enum _PRL_VM_ACTION_ON_STOP {
  PAS_KEEP_VM_WINDOW_OPEN = 0,
  PAS_CLOSE_VM_WINDOW = 1,
  PAS_QUIT_APPLICATION = 2
} PRL_VM_ACTION_ON_STOP;
```

File

PrlEnums.h

PRL_VM_ACTION_ON_WINDOW_CLOSE Enumeration

Action on VM when VM window gets closed

Syntax

```
typedef enum _PRL_VM_ACTION_ON_WINDOW_CLOSE {
  PWC_VM_UNKNOWN_ACTION = 0xffff,
  PWC_VM_STOP = 0,
  PWC_VM_SUSPEND = 1,
  PWC_VM_ASK_USER = 2,
  PWC_VM_PAUSE = 3,
  PWC_VM_SHUTDOWN = 4,
  PWC_VM_DO_NOTHING = 5
} PRL_VM_ACTION_ON_WINDOW_CLOSE;
```

File

PrlEnums.h

PRL_VM_AUTOSTART_OPTION Enumeration

VM autostart options

Syntax

```
typedef enum _PRL_VM_AUTOSTART_OPTION {
  PAO_VM_START_MANUAL = 0,
  PAO_VM_START_ON_LOAD = 1,
  PAO_VM_START_ON_RELOAD = 2,
```

```
  PAO_VM_START_ON_GUI_APP_STARTUP = 3,
  PAO_VM_START_ON_GUI_VM_WINDOW_OPEN = 4,
  PAO_VM_START_ON_GUI_WINDOW_LOAD = PAO_VM_START_ON_GUI_APP_STARTUP
} PRL_VM_AUTOSTART_OPTION;
```

File

PrlEnums.h

Members

PAO_VM_START_ON_GUI_WINDOW_LOAD

OBSOLETE CONSTANTS

PRL_VM_AUTOSTOP_OPTION Enumeration

VM autostop options

Syntax

```
typedef enum _PRL_VM_AUTOSTOP_OPTION {
  PAO_VM_STOP = 0,
  PAO_VM_SUSPEND = 1,
  PAO_VM_SHUTDOWN = 2
} PRL_VM_AUTOSTOP_OPTION;
```

File

PrlEnums.h

PRL_VM_BACKUP_FLAGS Enumeration

VM backup functionality flags set

Syntax

```
typedef enum _PRL_VM_BACKUP_FLAGS {
  PBSL_LOW_SECURITY = PSL_LOW_SECURITY,
  PBSL_NORMAL_SECURITY = PACF_NORMAL_SECURITY,
  PBSL_HIGH_SECURITY = PACF_HIGH_SECURITY,
  PBT_FULL = 1<<(PACF_MAX+1),
  PBT_INCREMENTAL = 1<<(PACF_MAX+2),
  PBT_DIFFERENTIAL = 1<<(PACF_MAX+3),
  PBT_RESTORE_TO_COPY = 1<<(PACF_MAX+4),
  PBT_IGNORE_NOT_EXISTS = 1<<(PACF_MAX+5),
  PBT_UNCOMPRESSED = 1<<(PACF_MAX+6),
  PBT_VM = 1<<(PACF_MAX+7),
  PBT_CT = 1<<(PACF_MAX+8),
  PBT_BACKUP_ID = 1<<(PACF_MAX+9),
  PBT_CHAIN = 1<<(PACF_MAX+10),
  PBT_KEEP_CHAIN = 1<<(PACF_MAX+11),
  PBT_DIRECT_DATA_CONNECTION = 1<<(PACF_MAX+12)
} PRL_VM_BACKUP_FLAGS;
```

File

PrlCommandsFlags.h

Type Parameters

PBT_FULL

full backup

PBT_INCREMENTAL
incremental backup

PBT_DIFFERENTAL
differental backup

PBT_RESTORE_TO_COPY
change Vm name/uuid/MAC address on restore

PBT_UNCOMPRESSED
uncompressed backup

PBT_VM
for virtual mashines

PBT_CT
for containers

PBT_BACKUP_ID
get information about a certain backup

PBT_CHAIN
get information about the whole backup chain for a certain backup

PBT_KEEP_CHAIN
preserve the rest of the backup chain on backup removal

PBT_DIRECT_DATA_CONNECTION
disables tunneling for backup (a laissez-fair data transfer)

PRL_VM_BACKUP_TYPE Enumeration

Hint about the type of VE inside a backup archive

Syntax

```
typedef enum _PRL_VM_BACKUP_TYPE {
  PVBT_VM = 0,
  PVBT_CT_VZFS = 1,
  PVBT_CT_PLOOP = 2,
  PVBT_CT_VZWIN = 3
} PRL_VM_BACKUP_TYPE;
```

File

PrlEnums.h

PRL_VM_BOOT_ORDER Enumeration

VM boot order

Syntax

```
typedef enum _PRL_VM_BOOT_ORDER {
  PBO_HDD_CD_FLOPPY = 0,
  PBO_FLOPPY_HDD_CD = 1,
  PBO_CD_HDD_FLOPPY = 2
} PRL_VM_BOOT_ORDER;
```

File

PrlEnums.h

PRL_VM_COLOR Enumeration

VM colors

Syntax

```
typedef enum _PRL_VM_COLOR {
  PVC_COLOR_RED = 0x00FA645A,
  PVC_COLOR_ORANGE = 0x00F6AA44,
  PVC_COLOR_YELLOW = 0x00EFDB47,
  PVC_COLOR_GREEN = 0x00B4D747,
  PVC_COLOR_BLUE = 0x005AA3FF,
  PVC_COLOR_PURPLE = 0x00C08ED8,
  PVC_COLOR_GREY = 0x00808080,
  PVC_NO_COLOR = 0x00000000
} PRL_VM_COLOR;
```

File

PrlEnums.h

PRL_VM_COMMIT_FLAGS Enumeration

PrlVm_CommitEx (p. 455) flags

Syntax

```
typedef enum _PRL_VM_COMMIT_FLAGS {
  PVCF_DESTROY_HDD_BUNDLE = 1<<(PACF_MAX+1),
  PVCF_WAIT_FOR_APPLY = 1<<(PACF_MAX+2),
  PEVF_WAIT_FOR_APPLY = PVCF_WAIT_FOR_APPLY,
  PVCF_DESTROY_HDD_BUNDLE_FORCE = 1<<(PACF_MAX+3),
  PVCF_DETACH_HDD_BUNDLE = 1<<(PACF_MAX+4),
  PVCF_RENAME_EXT_DISKS = 1<<(PACF_MAX+5)
} PRL_VM_COMMIT_FLAGS;
```

File

PrlCommandsFlags.h

Members

PVCF_DESTROY_HDD_BUNDLE
specifies to destroy HDD bundle physically on hard disk remove operation

PVCF_WAIT_FOR_APPLY
wait for apply

PEVF_WAIT_FOR_APPLY
to provide compatibility

PVCF_DESTROY_HDD_BUNDLE_FORCE
destroy HDD bundle physically even if used in snapshots

PRL_VM_CONFIG_SECTIONS Enumeration

VM configuration sections

Syntax

```
typedef enum _PRL_VM_CONFIG_SECTIONS {
  PVC_VALIDATE_CHANGES_ONLY = 0,
  PVC_ALL = 1,
  PVC_GENERAL_PARAMETERS = 2,
  PVC_BOOT_OPTION = 3,
  PVC_REMOTE_DISPLAY = 4,
  PVC_SHARED_FOLDERS = 5,
  PVC_CPU = 6,
  PVC_MAIN_MEMORY = 7,
  PVC_VIDEO_MEMORY = 8,
  PVC_FLOPPY_DISK = 9,
  PVC_CD_DVD_ROM = 10,
  PVC_HARD_DISK = 11,
  PVC_NETWORK_ADAPTER = 12,
  PVC_SOUND = 13,
  PVC_SERIAL_PORT = 14,
  PVC_PARALLEL_PORT = 15,
  PVC_IDE_DEVICES = 16,
  PVC_SCSI_DEVICES = 17,
  PVC_GENERIC_PCI = 18,
  PVC_OFFLINE_MANAGEMENT_SETTINGS = 19,
  PVC_LICENSE_RESTRICTIONS = 20,
  PVC_SATA_DEVICES = 21,
  PVC_LAST_SECTION = PVC_SATA_DEVICES
} PRL_VM_CONFIG_SECTIONS;
```

File

PrlEnums.h

PRL_VM_CPULIMIT_TYPE Enumeration

Syntax

```
typedef enum _PRL_VM_CPULIMIT_TYPE {
  PRL_VM_CPULIMIT_FULL = 0,
  PRL_VM_CPULIMIT_GUEST = 1
} PRL_VM_CPULIMIT_TYPE;
```

File

PrlEnums.h

PRL_VM_DEV_EMULATION_TYPE Enumeration

VM device emulation type

Syntax

```
typedef enum _PRL_VM_DEV_EMULATION_TYPE {
  PDT_ANY_TYPE = 0xffff,
  PDT_USE_REAL_DEVICE = 0,
  PDT_USE_IMAGE_FILE = 1,
  PDT_USE_OUTPUT_FILE = 2,
  PDT_USE_OTHER = 3,
  PDT_USE_DIRECT_ASSIGN = 4,
  PDT_USE_FILE_SYSTEM = 5,
  PDT_USE_TCP = 6,
  PDT_USE_UDP = 7,
  PDT_USE_REAL_FLOPPY = PDT_USE_REAL_DEVICE,
  PDT_USE_REAL_HDD = PDT_USE_REAL_DEVICE,
  PDT_USE_REAL_CDDVD = PDT_USE_REAL_DEVICE,
  PDT_USE_REAL_SERIAL_PORT = PDT_USE_REAL_DEVICE,
  PDT_USE_SERIAL_PORT_SOCKET_MODE = PDT_USE_OTHER,
  PDT_USE_SERIAL_PORT_TCP_MODE = PDT_USE_TCP,
  PDT_USE_SERIAL_PORT_UDP_MODE = PDT_USE_UDP,
  PDT_USE_REAL_PARALLEL_PORT = PDT_USE_REAL_DEVICE,
  PDT_USE_PARALLEL_PORT_PRINTER_MODE = PDT_USE_OTHER,
  PDT_USE_PARALLEL_PORT_PRINT_TO_PDF_MODE = PDT_USE_IMAGE_FILE,
  PDT_USE_HOST_ONLY_NETWORK = PDT_USE_REAL_DEVICE,
  PDT_USE_SHARED_NETWORK = PDT_USE_IMAGE_FILE,
  PDT_USE_BRIDGE_ETHERNET = PDT_USE_OUTPUT_FILE,
  PDT_USE_AC97_SOUND = PDT_USE_REAL_DEVICE,
  PDT_USE_CREATIVE_SB16_SOUND = PDT_USE_IMAGE_FILE,
  PDT_USE_REAL_USB_CONTROLLER = PDT_USE_REAL_DEVICE,
  PDT_USE_BOOTCAMP = PDT_USE_OTHER
} PRL_VM_DEV_EMULATION_TYPE;
```

File

PrlEnums.h

PRL_VM_EXEC_APP_FILE_DESCRIPTOR Enumeration

App execution result extraction moderators: PFD_STDOUT : request stdout of executed in guest OS app PFD_STDERR : request stderr of executed in guest OS app

Syntax

```
typedef enum _PRL_VM_EXEC_APP_FILE_DESCRIPTOR {
  PFD_STDOUT = 1<<(PACF_MAX+1),
  PFD_STDERR = 1<<(PACF_MAX+2),
  PFD_STDIN = 1<<(PACF_MAX+3),
  PFD_ALL = (PFD_STDOUT | PFD_STDERR | PFD_STDIN)
} PRL_VM_EXEC_APP_FILE_DESCRIPTOR;
```

File

PrlCommandsFlags.h

PRL_VM_EXEC_RUN_PROGRAM_MODES Enumeration

Run app in guest modes: PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY : just run program in guest OS and do not wait result PRPM_RUN_PROGRAM_IN_SHELL : run program via shell and in shell environment PRPM_RUN_PROGRAM_ENTER : run shell in guest

Syntax

```
typedef enum _PRL_VM_EXEC_RUN_PROGRAM_MODES {
  PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY = 1<<(PACF_MAX+4),
  PRPM_RUN_PROGRAM_IN_SHELL = 1<<(PACF_MAX+5),
  PRPM_RUN_PROGRAM_ENTER = 1<<(PACF_MAX+6)
} PRL_VM_EXEC_RUN_PROGRAM_MODES;
```

File

PrlCommandsFlags.h

PRL_VM_LOCATION Enumeration

VM files location set PVL_UNKNOWN : VM files location unknown (mostly for invalid VMs) PVL_LOCAL_FS : VM files placed on local file system PVL_REMOTE_FS : VM files placed on remote file system PVL_USB_DRIVE : VM files placed on external USB drive PVL_FIREWIRE_DRIVE : VM files placed on external FireWire drive

Syntax

```
typedef enum _PRL_VM_LOCATION {
  PVL_UNKNOWN = 0,
  PVL_LOCAL_FS = 1,
  PVL_REMOTE_FS = 2,
  PVL_USB_DRIVE = 3,
  PVL_FIREWIRE_DRIVE = 4
} PRL_VM_LOCATION;
```

File

PrlEnums.h

PRL_VM_MIGRATION_FLAGS Enumeration

VM migration functionality flags set

Syntax

```
typedef enum _PRL_VM_MIGRATION_FLAGS {
  PVMSL_LOW_SECURITY = PSL_LOW_SECURITY,
  PVMSL_NORMAL_SECURITY = PACF_NORMAL_SECURITY,
  PVMSL_HIGH_SECURITY = PACF_HIGH_SECURITY,
  PVMT_COLD_MIGRATION = 1<<(PACF_MAX+1),
  PVMT_WARM_MIGRATION = 1<<(PACF_MAX+2),
  PVMT_HOT_MIGRATION = 1<<(PACF_MAX+3),
  PVM_DONT_RESUME_VM = 1<<(PACF_MAX+4),
  PVMT_DONT_RESUME_VM = PVM_DONT_RESUME_VM,
  PVMT_CLONE_MODE = 1<<(PACF_MAX+5),
```

```
    PVMT_SWITCH_TEMPLATE = 1<<(PACF_MAX+6),
    PVMT_CHANGE_SID = 1<<(PACF_MAX+7),
    PVMT_IGNORE_EXISTING_BUNDLE = 1<<(PACF_MAX+8),
    PVMT_REMOVE_SOURCE_BUNDLE = 1<<(PACF_MAX+9),
    PVMT_SEND_DISK_MAP = 1<<(PACF_MAX+10),
    PVMT_UNCOMPRESSED = 1<<(PACF_MAX+11),
    PVMT_DIRECT_DATA_CONNECTION = 1<<(PACF_MAX+12),
    PVMT_DONT_CREATE_DISK = 1<<(PACF_MAX+13)
} PRL_VM_MIGRATION_FLAGS;
```

File

PrlCommandsFlags.h

Type Parameters

PVMT_COLD_MIGRATION

VM migration with interruption of VM services (VM will be stopped on migration time).

PVMT_WARM_MIGRATION

VM services will have quite long downtime during migration but won't be interrupted.

PVMT_HOT_MIGRATION

VM services will have short downtime during migration without any interruption.

PVMT_DONT_RESUME_VM

do not start running VM after migration

PVMT_CLONE_MODE

preserves source VM

PVMT_SWITCH_TEMPLATE

applable just in clone mode - lets to clone remotely VM to template

PVMT_CHANGE_SID

for Windows VMs/containers lets to change SID after remote clone

PVMT_IGNORE_EXISTING_BUNDLE

do not fail if Vm bundle exist. For clone Vm on mounted iSCSI storage

PVMT_UNCOMPRESSED

do not compress data during migration

PVMT_DIRECT_DATA_CONNECTION

do not use tunnel for migration

PVMT_DONT_CREATE_DISK

don't create disk file on target host

PRL_VM_MOUSE_CURSOR_STATE Enumeration

Syntax

```
typedef enum _PRL_VM_MOUSE_CURSOR_STATE {
  PTS_NO_CURSOR = 0,
  PTS_SLIDING_CURSOR = 1,
  PTS_MOVING_CURSOR = 2
} PRL_VM_MOUSE_CURSOR_STATE;
```

File

PrlEnums.h

PRL_VM_MOUSE_STATE Enumeration

Syntax

```
typedef enum _PRL_VM_MOUSE_STATE {
  PTS_RELATIVE_MOUSE = 0,
  PTS_SLIDING_MOUSE = 1,
  PTS_ABSOLUTE_MOUSE = 2
} PRL_VM_MOUSE_STATE;
```

File

PrlEnums.h

PRL_VM_NET_ADAPTER_TYPE Enumeration

Syntax

```
typedef enum _PRL_VM_NET_ADAPTER_TYPE {
  PNT_UNDEFINED = 0,
  PNT_RTL = 1,
  PNT_E1000 = 2,
  PNT_VIRTIO = 3,
  PNT_E1000E = 4
} PRL_VM_NET_ADAPTER_TYPE;
```

File

PrlEnums.h

PRL_VM_NET_ADAPTER_WIFI_RULE Enumeration

Syntax

```
typedef enum _PRL_VM_NET_ADAPTER_WIFI_RULE {
  PNWR_GENERATED_MAC = 0,
  PNWR_HOST_MAC = 1,
  PNWR_AUTO = 2
} PRL_VM_NET_ADAPTER_WIFI_RULE;
```

File

PrlEnums.h

PRL_VM_NORMAL_START_FLAGS Enumeration

VM normal-mode start flags: PSM_VM_START_IGNORE_HA_CLUSTER : do not register VM on HA cluster in any case (for

server mode only) PNSF_CT_SKIP_ARPDETECT : Ct specific parameter

Syntax

```
typedef enum _PRL_VM_NORMAL_START_FLAGS {
  PNSF_VM_START_WAIT = 1<<(PACF_MAX+2),
  PNSF_CT_SKIP_ARPDETECT = 1<<(PACF_MAX+3)
} PRL_VM_NORMAL_START_FLAGS;
```

File

PrlCommandsFlags.h

Members

PNSF_VM_START_WAIT

PNSF_VM_START_IGNORE_HA_CLUSTER = 1<<(PACF_MAX+1), This flag is unused

PRL_VM_ON_CRASH_ACTION Enumeration

Specifies an on-crash actions

Syntax

```
typedef enum _PRL_VM_ON_CRASH_ACTION {
  POCA_PAUSE = 0,
  POCA_RESTART = 1
} PRL_VM_ON_CRASH_ACTION;
```

File

PrlEnums.h

PRL_VM_ON_CRASH_OPTION Enumeration

Specifies an on-crash options

Syntax

```
typedef enum _PRL_VM_ON_CRASH_OPTION {
  POCO_DUMP = 1<<(PACF_MAX+1),
  POCO_REPORT = 1<<(PACF_MAX+2),
  POCO_FULL = POCO_DUMP|POCO_REPORT
} PRL_VM_ON_CRASH_OPTION;
```

File

PrlCommandsFlags.h

PRL_VM_PRIORITY Enumeration

Background/foreground VM processes priority

Syntax

```
typedef enum _PRL_VM_PRIORITY {
  PVR_PRIORITY_LOW = 0,
```

```
  PVR_PRIORITY_NORMAL = 1,
  PVR_PRIORITY_HIGH = 2
} PRL_VM_PRIORITY;
```

File

PrlEnums.h

PRL_VM_REMOTE_DISPLAY_MODE Enumeration

Virtuozzo remote desktop modes

Syntax

```
typedef enum _PRL_VM_REMOTE_DISPLAY_MODE {
  PRD_DISABLED = 0,
  PRD_AUTO,
  PRD_MANUAL
} PRL_VM_REMOTE_DISPLAY_MODE;
```

File

PrlEnums.h

PRL_VM_SMARTGUARD_SCHEMA Enumeration

SmartGuard schema

Syntax

```
typedef enum _PRL_VM_SMARTGUARD_SCHEMA {
  PVSS_OPTIMIZED_FOR_TIME_MACHINE = 1,
  PVSS_CUSTOM = 2
} PRL_VM_SMARTGUARD_SCHEMA;
```

File

PrlEnums.h

Members

PVSS_OPTIMIZED_FOR_TIME_MACHINE

Automatic schema for time machine

PVSS_CUSTOM

User defined schema

PRL_VM_START_LOGIN_MODE Enumeration

VM start login mode

Syntax

```
typedef enum _PRL_VM_START_LOGIN_MODE {
  PLM_START_ACCOUNT = 0,
  PLM_ROOT_ACCOUNT = 1,
```

```
  PLM_USER_ACCOUNT = 2,
  PLM_START_ACOUNT = PLM_START_ACCOUNT,
  PLM_ROOT_ACOUNT = PLM_ROOT_ACCOUNT,
  PLM_USER_ACOUNT = PLM_USER_ACCOUNT
} PRL_VM_START_LOGIN_MODE;
```

File

PrlEnums.h

Members

PLM_START_ACOUNT

Obsolete values for compatibility

PRL_VM_START_MODE Enumeration

VM start mode: PSM_VM_START : start VM PSM_VM_SAFE_START : start VM in the safe mode PSM_VM_START_FOR_COMPACT : start VM only for compacting disk operation

Syntax

```
typedef enum _PRL_VM_START_MODE {
  PSM_VM_START = 1<<(PACF_MAX+1),
  PSM_VM_SAFE_START = 1<<(PACF_MAX+2),
  PSM_VM_START_FOR_COMPACT = 1<<(PACF_MAX+3),
  PSM_VM_START_LAST_ITEM = PSM_VM_START_FOR_COMPACT
} PRL_VM_START_MODE;
```

File

PrlCommandsFlags.h

Members

PSM_VM_START_LAST_ITEM

PSM_VM_START_LAST_ITEM has to be equal the last mode

PRL_VM_STATISTICS_FLAGS Enumeration

PrlVm_GetStatisticsEx (p. 470) flags

Syntax

```
typedef enum _PRL_VM_STATISTICS_FLAGS {
  PVMSF_HOST_DISK_SPACE_USAGE_ONLY = 1<<(PACF_MAX+1),
  PVMSF_GUEST_OS_INFORMATION_ONLY = 1<<(PACF_MAX+2)
} PRL_VM_STATISTICS_FLAGS;
```

File

PrlCommandsFlags.h

PRL_VM_STOP_FLAGS Enumeration

Virtual machine shutdown options.

Syntax

```
typedef enum _PRL_VM_STOP_FLAGS {
  PSF_FORCE = 1<<(PACF_MAX+1),
  PSF_NOFORCE = 1<<(PACF_MAX+2)
} PRL_VM_STOP_FLAGS;
```

File

PrlCommandsFlags.h

Remarks

- PSF_FORCE : forcibly stop the virtual machine if shutdown timed out.
- PSF_NOFORCE : do nothing if the virtual machine shutdown timed out.

PRL_VM_STOP_MODE Enumeration

Vm stop modes

Syntax

```
typedef enum _PRL_VM_STOP_MODE {
  PSM_KILL = 0,
  PSM_SHUTDOWN = 1,
  PSM_ACPI = 2,
  PSM_LAST = PSM_ACPI
} PRL_VM_STOP_MODE;
```

File

PrlEnums.h

Members

PSM_LAST

Should equals to the last item

PRL_VM_STORE_VALUE_FLAGS Enumeration

Store value by specified keys on server options.

Syntax

```
typedef enum _PRL_VM_STORE_VALUE_FLAGS {
  PSV_CALC_BOOT_CAMP_SIZE = 1<<(PACF_MAX+1)
} PRL_VM_STORE_VALUE_FLAGS;
```

File

PrlCommandsFlags.h

Remarks

- PSV_CALC_BOOT_CAMP : calculate VM size with Boot Camp disks

PRL_VM_TOOLS_STATE Enumeration

Virtual machine tools states enumeration

Syntax

```
typedef enum _PRL_VM_TOOLS_STATE {
  PTS_UNKNOWN = 0,
  PTS_POSSIBLY_INSTALLED = PTS_UNKNOWN,
  PTS_NOT_INSTALLED = 1,
  PTS_INSTALLED = 2,
  PTS_OUTDATED = 3
} PRL_VM_TOOLS_STATE;
```

File

PrlEnums.h

PRL_VM_TYPE Enumeration

Syntax

```
typedef enum _PRL_VM_TYPE {
  PVT_VM = 0,
  PVT_CT = 1
} PRL_VM_TYPE;
```

File

PrlEnums.h

PRL_VM_TYPE_FLAGS Enumeration

Syntax

```
typedef enum _PRL_VM_TYPE_FLAGS {
  PVTF_VM = 1<<(PACF_MAX+1),
  PVTF_CT = 1<<(PACF_MAX+2)
} PRL_VM_TYPE_FLAGS;
```

File

PrlCommandsFlags.h

PRL_VM_WINDOW_MODE Enumeration

VM window mode

Syntax

```
typedef enum _PRL_VM_WINDOW_MODE {
  PWM_DEFAULT_WINDOW_MODE = 0,
```

```
    PWM_WINDOWED_WINDOW_MODE = 1,
    PWM_FULL_SCREEN_WINDOW_MODE = 2
} PRL_VM_WINDOW_MODE;
```

File

PrlEnums.h

Other PRL Structs

Enumerations

	Name	Description
	_PRL_ADAPTER_INDEX (p. 965)	Virtuozzo adapter index reserved values: PAI_INVALID_ADAPTER: Virtuozzo adapter in general is not available PAI_GENERATE_INDEX: calculate a new Virtuozzo adapter index
	_PRL_ALLOWED_VM_COMMAND (p. 965)	Allowed vm commands enumeration
	_PRL_API_COMMAND_FLAGS (p. 966)	Common operations flags set.
	_PRL_API_INIT_FLAGS (p. 967)	Virtuozzo API initialization flags: PrlApi_InitEx (p. 739) command flags set PAIF_USE_GRAPHIC_MODE : specifies create SDK session which has access to the graphic mode
	_PRL_APP_IN_DOCK_MODE (p. 971)	VM application in Dock mode
	_PRL_APPLIANCE_DOWNLOAD_STATUS (p. 972)	Appliance download status
	_PRL_APPLICATION_MODE (p. 972)	Virtuozzo application mode in which SDK should work PAM_SERVER : SDK should working at Virtuozzo Service application mode
	_PRL_BEGIN_VM_BACKUP_FLAGS (p. 972)	PrlVm_BeginBackup flags.
	_PRL_BIOS_TYPE (p. 973)	BIOS type
	_PRL_CLONE_VM_FLAGS (p. 973)	PrlVm_CloneEx (p. 451) command flags set PCVF_DETACH_EXTERNAL_VIRTUAL_HDD • to remove link to Virtual Hdd which located outside of VM.pvm directory from cloned vm configuration.
	_PRL_CLUSTERED_DEVICE_SUBTYPE (p. 974)	Clustered device subtype
	_PRL_COMMIT_ENCRYPTION_FLAGS (p. 974)	PrlVm_CommitEncryption (p. 454) command flags
	_PRL_COMPACT_MODE (p. 974)	Compact mode: PCM_COMPACT_WITH_HARD_DISKS_INFO : show question before compacting with size disks info PCM_HARD_DISKS_INFO : just get disk info PCM_FULL_CLEAN_UP_VM : compact all disks and delete all unneccessary files

	_PRL_CONVERT_VIRTUAL_DISK (p. 975)	Convert virtual disk • PCVD_CANCEL : cancel disk conversion • PCVD_TO_PLAIN_DISK : convert disk from expanding to plain disk • PCVD_TO_EXPANDING_DISK : convert disk from plain to expanding disk • PCVD_TO_SPLIT_DISK : convert disk from non-split to split disk • PCVD_TO_NON_SPLIT_DISK : convert disk from split to non-split disk • PCVD_MERGE_ALL_SNAPSHOTS : merge all disk snapshots
	_PRL_COPY_CT_TEMPLATE_FLAGS (p. 975)	copy of CT templates functionality flags set
	_PRL_CORNER_ACTIONS (p. 976)	Window corner actions
	_PRL_CPU_FEATURES_EX (p. 976)	
	_PRL_CPU_HVT (p. 976)	CPU mode
	_PRL_CPU_MODE (p. 977)	CPU mode
	_PRL_CPULIMIT_TYPE (p. 977)	
	_PRL_CPUS_COUNT (p. 977)	constant for unlimited CPU number
	_PRL_CPUUNITS (p. 978)	
	_PRL_CREATE_DISK_IMAGE_FLAGS (p. 978)	Create image flags
	_PRL_CREATE_SNAPSHOT_FLAGS (p. 978)	VM snapshot creation flags set
	_PRL_CT_CAPMASK (p. 979)	Capabilities list
	_PRL_CT_FEATURES (p. 980)	CT Features
	_PRL_CT_REINSTALL_OPTIONS (p. 980)	
	_PRL_CT_RESOURCE (p. 980)	
	_PRL_CT_TEMPLATE_TYPE (p. 981)	Container templates types
	_PRL_DATA_STATISTIC_SEGMENTS (p. 981)	Data statistic segments
	_PRL_DELETE_SNAPSHOT_FLAGS (p. 981)	• PrlVm_DeleteSnapshotEx flags PDSF_BACKUP - delete snapshot for backup
	_PRL_DEVICE_ACTION (p. 982)	Device in use Action
	_PRL_DEVICE_TYPE (p. 982)	Device type
	_PRL_DISPLAY_BUFFER_FORMAT (p. 983)	Format of display buffer
	_PRL_DISPLAY_CODEC_COMPRESSION (p. 983)	Obsolete enum.
	_PRL_DISPLAY_CODEC_QUALITY (p. 984)	Obsolete enum.
	_PRL_DISPLAY_CODEC_TYPE (p. 984)	Obsolete enum.
	_PRL_DISPLAY_CONFIG_MODE (p. 984)	Display(s) configuration mode
	_PRL_DISTRO_MEDIA_TYPE (p. 985)	
	_PRL_DOCK_ICON_TYPE (p. 985)	VM application icon type in Dock
	_PRL_END_VM_BACKUP_FLAGS (p. 986)	PrlVmBackup_Commit / PrlVmBackup_Rollback flags.

	_PRL_EVENT_ISSUER_TYPE (p. 986)	Event issuer type
	_PRL_EVENT_TYPE (p. 986)	Different event types are identified by these event codes.
	_PRL_EXTERNAL_VM_TYPE (p. 991)	External VM type for supporting native VM
	_PRL_FEATURES_MATRIX (p. 991)	Features matrix list
	_PRL_FILE_SYSTEM_ELEMENT_TYPE (p. 992)	File system element type
	_PRL_FILE_SYSTEM_FS_TYPE (p. 992)	
	_PRL_FILE_SYSTEM_TYPE (p. 992)	File system general type
	_PRL_FIREWALL_DIRECTION (p. 993)	Firewall direction
	_PRL_FIREWALL_POLICY (p. 993)	Firewall policy
	_PRL_GENERIC_DEVICE_STATE (p. 993)	Generic device state PGS_CONNECTED_TO_HOST : device is used by host PGS_CONNECTED_TO_VM : device is used by VM PGS_RESERVED : reserved value PGS_NON_CONTROLLED_USB : USB device is not controlled PGS_CONNECTING_TO_VM : device belong to VM, but prl_usb_dev still not started
	_PRL_GENERIC_PCI_DEVICE_CLASS (p. 994)	Generic PCI device class: PGD_PCI_NETWORK : generic PCI network device PGD_PCI_DISPLAY : generic PCI display device PGD_PCI_SOUND : generic PCI sound device PGD_PCI_OTHER : another generic PCI device
	_PRL_GET_SNAPSHOT_TREE_FLAGS (p. 994)	PrlVm_GetSnapshotsTreeEx (p. 468) flags
	_PRL_GET_VM_CONFIG_FLAGS (p. 994)	Find VM operations:
	_PRL_GET_VM_LIST_FLAGS (p. 995)	List operations: Uses for PrlSrv_GetVmListEx (p. 266) operation
	_PRL_GRACEFUL_SHUTDOWN_CAPABILITIES (p. 995)	
	_PRL_GRACEFUL_SHUTDOWN_KIND (p. 995)	Graceful Shutdown types
	_PRL_GUEST_DISTRO_VERSION_ENUM (p. 996)	Windows installation distributive type.
	_PRL_GUEST_OS_SUPPORT_TYPE (p. 996)	Specifies the type of guest OS's support
	_PRL_HANDLE_TYPE (p. 996)	The _PRL_HANDLE_TYPE enumeration defines handle types.
	_PRL_HARD_DISK_IFACE_MASK (p. 998)	Hard disk interfce mask enum
	_PRL_HARD_DISK_INTERNAL_FORMAT (p. 999)	Hard disk image type
	_PRL_HOST_OS_TYPE (p. 999)	Host OS type
	_PRL_HW_INFO_NET_ADAPTER_TYPE (p. 999)	Network adapter types in hardware info list
	_PRL_IMAGE_FORMAT (p. 1000)	
	_PRL_INSTALL_APPLIANCE_FLAGS (p. 1000)	Install appliance options • PIAF_CANCEL : cancel appliance installation • PIAF_FORCE : do not cancel installation and begin a new one • if the previous installation became corrupted outside. • PIAF_STOP : temporary stop appliance downloading
	_PRL_IOLIMIT_TYPE (p. 1000)	
	_PRL_IOPRIO (p. 1001)	

	_PRL_ISCSI_STORAGE_CMD_FLAGS (p. 1001)	iSCSI storage management flags
	_PRL_ISCSI_STORAGE_FSTYPE_FLAGS (p. 1001)	iSCSI storage filesystem type
	_PRL_JOB_OPERATION_CODE (p. 1002)	Job operation codes
	_PRL_JOB_STATUS (p. 1005)	Job status enumeration
	_PRL_KEY (p. 1005)	Virtuozzo virtual keycodes.
	_PRL_KEY_EVENT (p. 1008)	
	_PRL_LICENSE_RESTRICTION_KEY (p. 1009)	License restriction options
	_PRL_LOGIN_LOCAL_FLAGS (p. 1010)	PrlSrv_LoginLocal (p. 276) flags
	_PRL_MASS_STORAGE_INTERFACE_TYPE (p. 1011)	Mass storage device interface type
	_PRL_MEMGUARANTEE_TYPE (p. 1011)	
	_PRL_MESSAGE_TYPE_ENUM (p. 1011)	Message type enumeration.
	_PRL_MOBILE_ADVANCED_AUTH_MODE (p. 1012)	Authorization mode, when mobile client establishing remote access to the host.
	_PRL_MOUNT_VIRTUAL_DISK (p. 1012)	Mount virtual disk flags
	_PRL_MOUSE_BUTTON (p. 1012)	
	_PRL_NET_ADAPTER_EMULATED_TYPE (p. 1013)	Network adapter emulated type
	_PRL_NET_ADAPTER_TYPE (p. 1013)	
	_PRL_NET_VIRTUAL_NETWORK_TYPE (p. 1013)	Virtual Network type.
	_PRL_NETFILTER_MODE (p. 1014)	Netfilter modes list
	_PRL_NETWORK_RATE_MPU (p. 1014)	Special values of network packet MPU for packet rate limitation.
	_PRL_ONLINE_COMPACT_MODE (p. 1014)	Online compact mode for virtual machine disks
	_PRL_OPTIMIZE_MODIFIERS_MODE (p. 1015)	Determines the mode of modifiers handling inside the guest
	_PRL_PARAM_FIELD_DATA_TYPE (p. 1015)	Event parameters fields types
	_PRL_PORT_FORWARDING_TYPE (p. 1015)	Port forwarding type: PPF_TCP : TCP port forwarding PPF_UDP : UDP port forwarding
	_PRL_PRINTER_INTERFACE_TYPE (p. 1016)	Printer device interface type
	_PRL_PRIVATE_EVENT_TYPE (p. 1016)	Private event types.
	_PRL_PROBLEM_REPORT_FLAGS (p. 1018)	Virtuozzo problem report flags functionality: PPRF_ADD_CLIENT_PART : specifies to add client information to problem report PPRF_ADD_SERVER_PART : specifies to add server information to problem report PPRF_DO_NOT_CREATE_HOST_SCREENSHOT : specifies do not create and add host screenshot to generating problem report
	_PRL_PROBLEM_REPORT_REASON (p. 1019)	Problem report reason enumeration
	_PRL_PROBLEM_REPORT_SCHEME (p. 1019)	Problem report scheme enumaration
	_PRL_PROBLEM_REPORT_TYPE (p. 1020)	Problem report type enumeration
	_PRL_PROCESS_STATE_TYPE (p. 1020)	Process state element
	_PRL_PRODUCT_ENUM (p. 1021)	Virtuozzo license products enumeration.
	_PRL_PROXY_CONNECTION_STATUS (p. 1021)	Proxy connection status
	_PRL_QUERY_ELEMENT_FLAGS (p. 1021)	Query element flags

	_PRL_REGISTER_3PARTY_VM_FLAGS (p. 1022)	PrlSrv_Register3rdPartyVm (p. 285) command flags set
	_PRL_REGISTER_NEW_VM (p. 1022)	PrlVm_RegEx (p. 485) flags
	_PRL_REGISTER_VM_FLAGS (p. 1022)	PrlSrv_RegisterVmEx (p. 288) command flags set
	_PRL_RESIZE_IMAGE_FLAGS (p. 1023)	PrlVmDev_ResizeImage (p. 766) command flags set
	_PRL_RUNNING_TASK_TYPE_ENUM (p. 1023)	Running task type enumeration.
	_PRL_SCALE_VIEW_MODE (p. 1023)	Aspect ratio mode
	_PRL_SECURITY_LEVEL (p. 1024)	Security level enumeration.
	_PRL_SERIAL_PORT_SOCKET_OPERATION_MODE (p. 1024)	Serial port operation mode when on socket
	_PRL_SERVICE_STATUS_ENUM (p. 1024)	Virtuozzo Service statuses
	_PRL_SET_USERPASSWD_FLAGS (p. 1025)	PrlVm_SetUserPasswd (p. 490) flags
	_PRL_SOUND_CHANNEL_TYPE (p. 1025)	Sound channel type
	_PRL_SRV_SHUTDOWN_FLAGS (p. 1025)	PrlSrv_ShutdownEx (p. 298) flags
	_PRL_SWITCH_SNAPSHOT_FLAGS (p. 1026)	PrlVm_SwitchSnapshotEx flags
	_PRL_TASKBAR_STATES (p. 1026)	Taskbar states
	_PRL_TIS_RECORD_FIELD (p. 1027)	Virtuozzo Tools Information Service record fields
	_PRL_TIS_RECORD_STATE (p. 1027)	Virtuozzo Tools Information Service record states
	_PRL_TOOLS_SIA_OPEN_IN_TYPES (p. 1027)	Shared Internet Applications 'open in' type PSIA_OPEN_DEFAULT : Open host link with host app; Open guest link with guest app PSIA_OPEN_IN_GUEST : Open all links with guest apps; PSIA_OPEN_IN_HOST : Open all links with host apps;
	_PRL_TOOLS_UTILITY_COMMAND_FLAG (p. 1028)	Virtuozzo Tools Utility command flags
	_PRL_TOOLS_UTILITY_COMMAND_ID (p. 1028)	Virtuozzo Tools Utility commands identifiers
	_PRL_UNDO_DISKS_MODE (p. 1029)	VM undo disks mode: PUD_DISABLE_UNDO_DISKS : disbale undo disks PUD_REVERSE_CHANGES : at stopping VM revert all disk changes PUD_COMMIT_CHANGES : at stopping VM apply all disk changes PUD_PROMPT_BEHAVIOUR : ask user which mode reverse or commit will be applied
	_PRL_UPDATE_LICENSE_FLAGS (p. 1029)	Update/Install license operations: Uses for PrlSrv_UpdateLicenseEx (p. 310) operation
	_PRL_USB_DEVICE_AUTO_CONNECT_OPTION (p. 1029)	USB autoconnect devices options
	_PRL_USB_DEVICE_CONNECT_REASON (p. 1030)	USB device connect/disconnect reason
	_PRL_USB_DEVICE_TYPE (p. 1030)	USB device type
	_PRL_VCMMD_COMMAND_FLAGS (p. 1032)	
	_PRL_VERSION_ENUM (p. 1032)	Virtuozzo license versions enumeration.
	_PRL_VIDEO_3D_ACCELERATION (p. 1032)	Video 3D acceleration
	_PRL_VIRTUAL_MACHINE_PROFILE (p. 1033)	Virtual machine profiles enum
	_PRL_VIRTUAL_NETWORK_FLAGS (p. 1033)	PrlSrv_AddVirtualNetwork (p. 224) flangs
	_PRL_VM_ON_CRASH_ACTION (p. 1033)	Specifies an on-crash actions
	_PRL_VM_ON_CRASH_OPTION (p. 1033)	Specifies an on-crash options
	_PRL_VMCTG_EVENT (p. 1034)	Tools events, passed to the callback (see PRL_VMCTG_TOOL_DESC and PRL_TOOL_HANDLER_PTR above)
	_PRL_VTD_STATUS_CODE (p. 1034)	VTD status code
	_PRL_WINDOW_CORNERS (p. 1035)	Window corners

	_PRL_WINDOW_EDGE_MASK (p. 1035)	Edges where taskbar can be placed (in coherence mode).
	PRL_ADAPTER_INDEX (p. 1036)	Virtuozzo adapter index reserved values: PAI_INVALID_ADAPTER: Virtuozzo adapter in general is not available PAI_GENERATE_INDEX: calculate a new Virtuozzo adapter index
	PRL_ALLOWED_VM_COMMAND (p. 1036)	Allowed vm commands enumeration
	PRL_API_COMMAND_FLAGS (p. 1037)	Common operations flags set.
	PRL_API_INIT_FLAGS (p. 1038)	Virtuozzo API initialization flags: PrlApi_InitEx (p. 739) command flags set PAIF_USE_GRAPHIC_MODE : specifies create SDK session which has access to the graphic mode
	PRL_APP_IN_DOCK_MODE (p. 1038)	VM application in Dock mode
	PRL_APPLIANCE_DOWNLOAD_STATUS (p. 1039)	Appliance download status
	PRL_APPLICATION_MODE (p. 1039)	Virtuozzo application mode in which SDK should work PAM_SERVER : SDK should working at Virtuozzo Service application mode
	PRL_BEGIN_VM_BACKUP_FLAGS (p. 1039)	PrlVm_BeginBackup flags.
	PRL_BIOS_TYPE (p. 1040)	BIOS type
	PRL_CLONE_VM_FLAGS (p. 1040)	PrlVm_CloneEx (p. 451) command flags set PCVF_DETACH_EXTERNAL_VIRTUAL_HDD • to remove link to Virtual Hdd which located outside of VM.pvm directory from cloned vm configuration.
	PRL_CLUSTERED_DEVICE_SUBTYPE (p. 1041)	Clustered device subtype
	PRL_COMMIT_ENCRYPTION_FLAGS (p. 1041)	PrlVm_CommitEncryption (p. 454) command flags
	PRL_COMPACT_MODE (p. 1041)	Compact mode: PCM_COMPACT_WITH_HARD_DISKS_INFO : show question before compacting with size disks info PCM_HARD_DISKS_INFO : just get disk info PCM_FULL_CLEAN_UP_VM : compact all disks and delete all unneccessary files
	PRL_CONVERT_VIRTUAL_DISK (p. 1042)	Convert virtual disk • PCVD_CANCEL : cancel disk conversion • PCVD_TO_PLAIN_DISK : convert disk from expanding to plain disk • PCVD_TO_EXPANDING_DISK : convert disk from plain to expanding disk • PCVD_TO_SPLIT_DISK : convert disk from non-split to split disk • PCVD_TO_NON_SPLIT_DISK : convert disk from split to non-split disk • PCVD_MERGE_ALL_SNAPSHOTS : merge all disk snapshots
	PRL_COPY_CT_TEMPLATE_FLAGS (p. 1042)	copy of CT templates functionality flags set
	PRL_CORNER_ACTIONS (p. 1043)	Window corner actions
	PRL_CPU_FEATURES_EX (p. 1043)	
	PRL_CPU_HVT (p. 1043)	CPU mode
	PRL_CPU_MODE (p. 1044)	CPU mode
	PRL_CPULIMIT_TYPE (p. 1044)	

	PRL_CPUS_COUNT (p. 1044)	constant for unlimited CPU number
	PRL_CPUUITS (p. 1045)	
	PRL_CREATE_DISK_IMAGE_FLAGS (p. 1045)	Create image flags
	PRL_CREATE_SNAPSHOT_FLAGS (p. 1045)	VM snapshot creation flags set
	PRL_CT_CAPMASK (p. 1046)	Capabilities list
	PRL_CT_FEATURES (p. 1047)	CT Features
	PRL_CT_REINSTALL_OPTIONS (p. 1047)	
	PRL_CT_RESOURCE (p. 1047)	
	PRL_CT_TEMPLATE_TYPE (p. 1048)	Container templates types
	PRL_DATA_STATISTIC_SEGMENTS (p. 1048)	Data statistic segments
	PRL_DELETE_SNAPSHOT_FLAGS (p. 1048)	• PrlVm_DeleteSnapshotEx flags PDSF_BACKUP - delete snapshot for backup
	PRL_DEVICE_ACTION (p. 1049)	Device in use Action
	PRL_DEVICE_TYPE (p. 1049)	Device type
	PRL_DISPLAY_BUFFER_FORMAT (p. 1050)	Format of display buffer
	PRL_DISPLAY_CODEC_COMPRESSION (p. 1051)	Obsolete enum.
	PRL_DISPLAY_CODEC_QUALITY (p. 1051)	Obsolete enum.
	PRL_DISPLAY_CODEC_TYPE (p. 1051)	Obsolete enum.
	PRL_DISPLAY_CONFIG_MODE (p. 1052)	Display(s) configuration mode
	PRL_DISTRO_MEDIA_TYPE (p. 1052)	
	PRL_DOCK_ICON_TYPE (p. 1053)	VM application icon type in Dock
	PRL_END_VM_BACKUP_FLAGS (p. 1053)	PrlVmBackup_Commit / PrlVmBackup_Rollback flags.
	PRL_EVENT_ISSUER_TYPE (p. 1053)	Event issuer type
	PRL_EVENT_TYPE (p. 1054)	Different event types are identified by these event codes.
	PRL_EXTERNAL_VM_TYPE (p. 1058)	External VM type for supporting native VM
	PRL_FEATURES_MATRIX (p. 1058)	Features matrix list
	PRL_FILE_SYSTEM_ELEMENT_TYPE (p. 1059)	File system element type
	PRL_FILE_SYSTEM_FS_TYPE (p. 1059)	
	PRL_FILE_SYSTEM_TYPE (p. 1060)	File system general type
	PRL_FIREWALL_DIRECTION (p. 1060)	Firewall direction
	PRL_FIREWALL_POLICY (p. 1060)	Firewall policy
	PRL_GENERIC_DEVICE_STATE (p. 1061)	Generic device state PGS_CONNECTED_TO_HOST : device is used by host PGS_CONNECTED_TO_VM : device is used by VM PGS_RESERVED : reserved value PGS_NON_CONTROLLED_USB : USB device is not controlled PGS_CONNECTING_TO_VM : device belong to VM, but prl_usb_dev still not started
	PRL_GENERIC_PCI_DEVICE_CLASS (p. 1061)	Generic PCI device class: PGD_PCI_NETWORK : generic PCI network device PGD_PCI_DISPLAY : generic PCI display device PGD_PCI_SOUND : generic PCI sound device PGD_PCI_OTHER : another generic PCI device
	PRL_GET_SNAPSHOT_TREE_FLAGS (p. 1061)	PrlVm_GetSnapshotsTreeEx (p. 468) flags
	PRL_GET_VM_CONFIG_FLAGS (p. 1062)	Find VM operations:
	PRL_GET_VM_LIST_FLAGS (p. 1062)	List operations: Uses for PrlSrv_GetVmListEx (p. 266) operation
	PRL_GRACEFUL_SHUTDOWN_CAPABILITIES (p. 1062)	

	PRL_GRACEFUL_SHUTDOWN_KIND (p. 1063)	Graceful Shutdown types
	PRL_GUEST_DISTRO_VERSION_ENUM (p. 1063)	Windows installation distributive type.
	PRL_GUEST_OS_SUPPORT_TYPE (p. 1063)	Specifies the type of guest OS's support
	PRL_HANDLE_TYPE (p. 1064)	The _PRL_HANDLE_TYPE (p. 996) enumeration defines handle types.
	PRL_HARD_DISK_IFACE_MASK (p. 1065)	Hard disk interfce mask enum
	PRL_HARD_DISK_INTERNAL_FORMAT (p. 1066)	Hard disk image type
	PRL_HOST_OS_TYPE (p. 1066)	Host OS type
	PRL_HW_INFO_NET_ADAPTER_TYPE (p. 1067)	Network adapter types in hardware info list
	PRL_IMAGE_FORMAT (p. 1067)	
	PRL_INSTALL_APPLIANCE_FLAGS (p. 1068)	Install appliance options • PIAF_CANCEL : cancel appliance installation • PIAF_FORCE : do not cancel installation and begin a new one • if the previous installation became corrupted outside. • PIAF_STOP : temporary stop appliance downloading
	PRL_IOPRIO (p. 1068)	
	PRL_ISCSI_STORAGE_CMD_FLAGS (p. 1068)	iSCSI storage management flags
	PRL_ISCSI_STORAGE_FSTYPE_FLAGS (p. 1069)	iSCSI storage filesystem type
	PRL_JOB_OPERATION_CODE (p. 1069)	Job operation codes
	PRL_JOB_STATUS (p. 1072)	Job status enumeration
	PRL_KEY (p. 1073)	Virtuozzo virtual keycodes.
	PRL_KEY_EVENT (p. 1076)	
	PRL_LICENSE_RESTRICTION_KEY (p. 1076)	License restriction options
	PRL_LOGIN_LOCAL_FLAGS (p. 1078)	PrlSrv_LoginLocal (p. 276) flags
	PRL_MASS_STORAGE_INTERFACE_TYPE (p. 1078)	Mass storage device interface type
	PRL_MEMGUARANTEE_TYPE (p. 1079)	
	PRL_MESSAGE_TYPE_ENUM (p. 1079)	Message type enumeration.
	PRL_MOBILE_ADVANCED_AUTH_MODE (p. 1079)	Authorization mode, when mobile client establishing remote access to the host.
	PRL_MOUNT_VIRTUAL_DISK (p. 1079)	Mount virtual disk flags
	PRL_MOUSE_BUTTON (p. 1080)	
	PRL_NET_ADAPTER_EMULATED_TYPE (p. 1080)	Network adapter emulated type
	PRL_NET_ADAPTER_TYPE (p. 1081)	
	PRL_NET_VIRTUAL_NETWORK_TYPE (p. 1081)	Virtual Network type.
	PRL_NETFILTER_MODE (p. 1081)	Netfilter modes list
	PRL_NETWORK_RATE_MPU (p. 1082)	Special values of network packet MPU for packet rate limitation.
	PRL_ONLINE_COMPACT_MODE (p. 1082)	Online compact mode for virtual machine disks
	PRL_OPTIMIZE_MODIFIERS_MODE (p. 1082)	Determines the mode of modifiers handling inside the guest
	PRL_PARAM_FIELD_DATA_TYPE (p. 1083)	Event parameters fields types
	PRL_PORT_FORWARDING_TYPE (p. 1083)	Port forwarding type: PPF_TCP : TCP port forwarding PPF_UDP : UDP port forwarding
	PRL_PRINTER_INTERFACE_TYPE (p. 1083)	Printer device interface type
	PRL_PRIVATE_EVENT_TYPE (p. 1084)	Private event types.

	PRL_PROBLEM_REPORT_FLAGS (p. 1086)	Virtuozzo problem report flags functionality: PPRF_ADD_CLIENT_PART : specifies to add client information to problem report PPRF_ADD_SERVER_PART : specifies to add server information to problem report PPRF_DO_NOT_CREATE_HOST_SCREENSHOT : specifies do not create and add host screenshot to generating problem report
	PRL_PROBLEM_REPORT_REASON (p. 1086)	Problem report reason enumeration
	PRL_PROBLEM_REPORT_SCHEME (p. 1087)	Problem report scheme enumaration
	PRL_PROBLEM_REPORT_TYPE (p. 1087)	Problem report type enumeration
	PRL_PROCESS_STATE_TYPE (p. 1088)	Process state element
	PRL_PRODUCT_ENUM (p. 1088)	Virtuozzo license products enumeration.
	PRL_PROXY_CONNECTION_STATUS (p. 1089)	Proxy connection status
	PRL_QUERY_ELEMENT_FLAGS (p. 1089)	Query element flags
	PRL_REGISTER_3PARTY_VM_FLAGS (p. 1089)	PrlSrv_Register3rdPartyVm (p. 285) command flags set
	PRL_REGISTER_NEW_VM (p. 1089)	PrlVm_RegEx (p. 485) flags
	PRL_REGISTER_VM_FLAGS (p. 1090)	PrlSrv_RegisterVmEx (p. 288) command flags set
	PRL_RESIZE_IMAGE_FLAGS (p. 1090)	PrlVmDev_ResizeImage (p. 766) command flags set
	PRL_RUNNING_TASK_TYPE_ENUM (p. 1090)	Running task type enumeration.
	PRL_SCALE_VIEW_MODE (p. 1091)	Aspect ratio mode
	PRL_SECURITY_LEVEL (p. 1091)	Security level enumeration.
	PRL_SERIAL_PORT_SOCKET_OPERATION_MODE (p. 1092)	Serial port operation mode when on socket
	PRL_SERVICE_STATUS_ENUM (p. 1092)	Virtuozzo Service statuses
	PRL_SET_USERPASSWD_FLAGS (p. 1092)	PrlVm_SetUserPasswd (p. 490) flags
	PRL_SOUND_CHANNEL_TYPE (p. 1093)	Sound channel type
	PRL_SRV_SHUTDOWN_FLAGS (p. 1093)	PrlSrv_ShutdownEx (p. 298) flags
	PRL_SWITCH_SNAPSHOT_FLAGS (p. 1094)	PrlVm_SwitchSnapshotEx flags
	PRL_TASKBAR_STATES (p. 1094)	Taskbar states
	PRL_TIS_RECORD_FIELD (p. 1094)	Virtuozzo Tools Information Service record fields
	PRL_TIS_RECORD_STATE (p. 1095)	Virtuozzo Tools Information Service record states
	PRL_TOOLS_SIA_OPEN_IN_TYPES (p. 1095)	Shared Internet Applications 'open in' type PSIA_OPEN_DEFAULT : Open host link with host app; Open guest link with guest app PSIA_OPEN_IN_GUEST : Open all links with guest apps; PSIA_OPEN_IN_HOST : Open all links with host apps;
	PRL_TOOLS_UTILITY_COMMAND_FLAG (p. 1095)	Virtuozzo Tools Utility command flags
	PRL_TOOLS_UTILITY_COMMAND_ID (p. 1096)	Virtuozzo Tools Utility commands identifiers
	PRL_UNDO_DISKS_MODE (p. 1096)	VM undo disks mode: PUD_DISABLE_UNDO_DISKS : disbale undo disks PUD_REVERSE_CHANGES : at stopping VM revert all disk changes PUD_COMMIT_CHANGES : at stopping VM apply all disk changes PUD_PROMPT_BEHAVIOUR : ask user which mode reverse or commit will be applied
	PRL_UPDATE_LICENSE_FLAGS (p. 1097)	Update/Install license operations: Uses for PrlSrv_UpdateLicenseEx (p. 310) operation
	PRL_USB_DEVICE_AUTO_CONNECT_OPTION (p. 1097)	USB autoconnect devices options
	PRL_USB_DEVICE_CONNECT_REASON (p. 1097)	USB device connect/disconnect reason

	PRL_USB_DEVICE_TYPE (p. 1098)	USB device type
	PRL_VCMMD_COMMAND_FLAGS (p. 1099)	
	PRL_VERSION_ENUM (p. 1099)	Virtuozzo license versions enumeration.
	PRL_VIDEO_3D_ACCELERATION (p. 1100)	Video 3D acceleration
	PRL_VIDEO_RECEIVER_CONNECTION_STATE (p. 1100)	Streaming connection status enum
	PRL_VIRTUAL_MACHINE_PROFILE (p. 1100)	Virtual machine profiles enum
	PRL_VIRTUAL_NETWORK_FLAGS (p. 1101)	PrlSrv_AddVirtualNetwork (p. 224) flangs
	PRL_VMCTG_EVENT (p. 1101)	Tools events, passed to the callback (see PRL_VMCTG_TOOL_DESC and PRL_TOOL_HANDLER_PTR above)
	PRL_VTD_STATUS_CODE (p. 1101)	VTD status code
	PRL_WINDOW_CORNERS (p. 1102)	Window corners
	PRL_WINDOW_EDGE_MASK (p. 1102)	Edges where taskbar can be placed (in coherence mode).

Structures

	Name	Description
	__PRL_DISK_PARAMETERS (p. 963)	Virtuozzo disk parameters.
	__PRL_IMAGE_PARAMETERS (p. 964)	Virtuozzo storage parameters
	__PRL_SNAPSHOT_ELEMENT (p. 964)	Virtuozzo snapshot tree element
	_IPluginInfo (p. 964)	Generic information about plugin
	_PRL_API_METHOD (p. 967)	Virtuozzo API method
	_PRL_CHS (p. 973)	CHS format of disk
	_PrlPlugin (p. 1035)	Base interface
	PRL_CHS (p. 1040)	CHS format of disk
	PRL_DISK_PARAMETERS (p. 1050)	Virtuozzo disk parameters.
	PRL_IMAGE_PARAMETERS (p. 1067)	Virtuozzo storage parameters
	PRL_SNAPSHOT_ELEMENT (p. 1093)	Virtuozzo snapshot tree element

__PRL_DISK_PARAMETERS Structure

Virtuozzo disk parameters.

Syntax

```
struct __PRL_DISK_PARAMETERS {
  PRL_UINT32 m_Heads;
  PRL_UINT32 m_Cylinders;
  PRL_UINT32 m_Sectors;
  PRL_UINT64 m_SizeInSectors;
  PRL_UINT32 m_BlockSize;
  PRL_UINT32 m_Parts;
  PRL_IMAGE_PARAMETERS_PTR m_Storages;
  PRL_GUID Uid;
  PRL_CHAR_PTR Name;
```

```
};
```

File

PrlDisk.h

Members

m_Storages

Following parameters are available only at SDK v3.

Remarks

Differs from internal due to lists classes

__PRL_IMAGE_PARAMETERS Structure

Virtuozzo storage parameters

Syntax

```
struct __PRL_IMAGE_PARAMETERS {
  PRL_IMAGE_TYPE Type;
  PRL_UINT64 uStart;
  PRL_UINT64 uSize;
  PRL_CHAR_PTR pFileName;
};
```

File

PrlDisk.h

__PRL_SNAPSHOT_ELEMENT Structure

Virtuozzo snapshot tree element

Syntax

```
struct __PRL_SNAPSHOT_ELEMENT {
  PRL_GUID m_GUID;
  struct __PRL_SNAPSHOT_ELEMENT* m_pNext;
  struct __PRL_SNAPSHOT_ELEMENT* m_pChild;
};
```

File

PrlDisk.h

_IPluginInfo Structure

Generic information about plugin

Syntax

```
struct _IPluginInfo {
  PRL_CONST_STR Vendor;
  PRL_CONST_STR Copyright;
```

```
    PRL_CONST_STR DescShort;
    PRL_CONST_STR DescLong;
    PRL_UINT32 Major;
    PRL_UINT32 Minor;
    PRL_UINT32 Build;
    PRL_GUID Type;
  };
```

File

PrlPluginClasses.h

_PRL_ADAPTER_INDEX Enumeration

Virtuozzo adapter index reserved values: PAI_INVALID_ADAPTER: Virtuozzo adapter in general is not available PAI_GENERATE_INDEX: calculate a new Virtuozzo adapter index

Syntax

```
enum _PRL_ADAPTER_INDEX {
  PAI_INVALID_ADAPTER = -1,
  PAI_GENERATE_INDEX = -2
};
```

File

PrlEnums.h

_PRL_ALLOWED_VM_COMMAND Enumeration

Allowed vm commands enumeration

Syntax

```
enum _PRL_ALLOWED_VM_COMMAND {
  PAR_VM_START_ACCESS = 0,
  PAR_VM_STOP_ACCESS = 1,
  PAR_VM_PAUSE_ACCESS = 2,
  PAR_VM_RESET_ACCESS = 3,
  PAR_VM_SUSPEND_ACCESS = 4,
  PAR_VM_RESUME_ACCESS = 5,
  PAR_VM_DROPSUSPENDEDSTATE_ACCESS = 6,
  PAR_VM_CLONE_ACCESS = 7,
  PAR_VM_DELETE_ACCESS = 8,
  PAR_VM_GETPROBLEMREPORT_ACCESS = 9,
  PAR_VM_GETCONFIG_ACCESS = 10,
  PAR_VM_GETSTATISTICS_ACCESS = 11,
  PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS = 12,
  PAR_VM_SUBSCRIBETOGUESTSTATISTICS_ACCESS = PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS,
  PAR_VM_UNSUBSCRIBEFROMGUESTSTATISTICS_ACCESS = PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS,
  PAR_VM_UNREG_ACCESS = 13,
  PAR_VM_COMMIT_ACCESS = 14,
  PAR_VM_EDITING_ACCESS = PAR_VM_COMMIT_ACCESS,
  PAR_VMDEV_CONNECT_ACCESS = 15,
  PAR_VMDEV_DISCONNECT_ACCESS = 16,
  PAR_VMDEV_CREATEIMAGE_ACCESS = 17,
  PAR_VM_SEND_ANSWER_ACCESS = 18,
  PAR_VM_GET_VMINFO_ACCESS = 19,
  PAR_VM_INSTALL_TOOLS_ACCESS = 20,
```

```
    PAR_VM_INITIATE_DEV_STATE_NOTIFICATIONS_ACCESS = 21,
    PAR_VM_UPDATE_SECURITY_ACCESS = 22,
    PAR_VM_CANCEL_COMPACT_ACCESS = 23,
    PAR_VM_REGISTER_ACCESS = 24,
    PAR_VM_MIGRATE_ACCESS = 25,
    PAR_VM_START_STOP_VNC_SERVER = 26,
    PAR_VM_GET_SUSPENDED_SCREEN_ACCESS = 27,
    PAR_VM_PERFSTAT_ACCESS = 28,
    PAR_VM_LOCK_ACCESS = 29,
    PAR_VM_MIGRATE_CANCEL_ACCESS = PAR_VM_MIGRATE_ACCESS,
    PAR_VM_RUN_COMPRESSOR_ACCESS = 31,
    PAR_VM_CANCEL_COMPRESSOR_ACCESS = 32,
    PAR_VM_CREATE_SNAPSHOT_ACCESS = 33,
    PAR_VM_SWITCH_TO_SNAPSHOT_ACCESS = 34,
    PAR_VM_DELETE_SNAPSHOT_ACCESS = 35,
    PAR_VM_START_EX_ACCESS = 36,
    PAR_VM_UNLOCK_ACCESS = 37,
    PAR_VM_RESTART_GUEST_ACCESS = 38,
    PAR_VM_INSTALL_UTILITY_ACCESS = 39,
    PAR_VM_UPDATE_TOOLS_SECTION = 40,
    PAR_VM_RESTORE_ACCESS = 41,
    PAR_VM_CREATE_BACKUP_ACCESS = 42,
    PAR_VM_RESIZE_DISK_ACCESS = 43,
    PAR_VM_CREATE_ACCESS = 44,
    PAR_SRV_USER_PROFILE_BEGINEDIT_ACCESS = 45,
    PAR_SRV_USER_PROFILE_COMMIT_ACCESS = 46,
    PAR_SRV_SERVER_PROFILE_BEGINEDIT_ACCESS = 47,
    PAR_SRV_SERVER_PROFILE_COMMIT_ACCESS = 48,
    PAR_VM_GUI_VIEW_MODE_CHANGE_ACCESS = 49,
    PAR_VM_BEGINEDIT_ACCESS = 50,
    PAR_VM_RESTORE_BACKUP_ACCESS = 51,
    PAR_VM_COMPACT_ACCESS = 52,
    PAR_VM_CHANGE_SID_ACCESS = 53,
    PAR_VM_RESET_UPTIME_ACCESS = 54,
    PAR_VM_INTERNAL_CMD_ACCESS = 55,
    PAR_VM_GET_VIRT_DEV_INFO = 56,
    PAR_VM_GET_TOOLS_INFO = 57,
    PAR_VM_CONVERT_DISKS_ACCESS = 58,
    PAR_VM_ENCRYPT_OPERATION_ACCESS = 59,
    PAR_VM_CHANGE_PASSWORD_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_ENCRYPT_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_DECRYPT_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_SET_PROTECTION_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_REMOVE_PROTECTION_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_AUTHORISE_ACCESS = 60,
    PAR_VMDEV_COPY_IMAGE_ACCESS = 61,
    PAR_VM_MOUNT_ACCESS = 62,
    PAR_VM_MOVE_ACCESS = 63,
    PAR_VM_CHANGE_GUEST_OS_PASSWORD_ACCESS = 64,
    PAR_SRV_SENDPROBLEMREPORT_ACCESS = 65,
    PAR_MAX
  };
```

File

PrlEnums.h

Members

PAR_MAX
should be the last

_PRL_API_COMMAND_FLAGS Enumeration

Common operations flags set.

Syntax

```
enum _PRL_API_COMMAND_FLAGS {
  PACF_NORMAL_SECURITY = PSL_NORMAL_SECURITY,
  PACF_HIGH_SECURITY = PSL_HIGH_SECURITY,
  PACF_NON_INTERACTIVE_MODE = 1<<2,
  PACF_CANCEL_TASK_ON_END_SESSION = 1<<3,
  PACF_MAX = 10,
  PACF_LAST = 1<<PACF_MAX
};
```

File

PrlCommandsFlags.h

Members

PACF_NORMAL_SECURITY
Client connection security mode

PACF_NON_INTERACTIVE_MODE
Session interactive mode

PACF_CANCEL_TASK_ON_END_SESSION
Cancel long-term operation by session disconnect

PACF_MAX
Interval from here to PACF_LAST reserved for further purposes.

Remarks

Please do not override values

_PRL_API_INIT_FLAGS Enumeration

Virtuozzo API initialization flags: PrlApi_InitEx (p. 739) command flags set PAIF_USE_GRAPHIC_MODE : specifies create SDK session which has access to the graphic mode

Syntax

```
enum _PRL_API_INIT_FLAGS {
  PAIF_USE_GRAPHIC_MODE = 1<<(PACF_MAX+1),
  PAIF_INIT_AS_APPSTORE_CLIENT = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

_PRL_API_METHOD Structure

Virtuozzo API method

Syntax

```
struct _PRL_API_METHOD {
  PRL_CONST_STR MethodName;
  PRL_UINT32 MethodAPIVersion;
  PRL_VOID_PTR MethodPtr;
  PRL_BOOL MethodIsDeprecated;
  PRL_BOOL MethodIsAsync;
  PRL_BOOL MethodIsSrv;
  struct _PRL_GUID {
    PRL_UINT32 Data1;
    PRL_UINT16 Data2;
    PRL_UINT16 Data3;
    PRL_UINT8 Data4[8];
  };
};
```

File

PrlTypes.h

Members

_PRL_API_METHOD Data Members

	Name	Description
	MethodAPIVersion (p. 969)	
	MethodIsAsync (p. 969)	
	MethodIsDeprecated (p. 969)	
	MethodIsSrv (p. 969)	
	MethodName (p. 969)	
	MethodPtr (p. 970)	

_PRL_API_METHOD Methods

	Name	Description
	PRL_RESULT (p. 970)	All events (either commands, input/output, etc.) are recieved by the client in this 'callback' functions, provided by the client to the library.

_PRL_API_METHOD Nested Types

Name	Description
PRL_API_METHOD_PTR (p. 970)	
PRL_CONST_API_METHOD_PTR (p. 970)	
PRL_CONST_GUID_PTR (p. 970)	
PRL_FILE_DESC (p. 970)	File descriptor type
PRL_GUID (p. 970)	Global unique identifier
PRL_GUID_PTR (p. 971)	
PRL_RESULT (p. 971)	Unified result error code for each and every method
PRL_RESULT_PTR (p. 971)	

_PRL_API_METHOD Structures

	Name	Description
	_PRL_GUID (p. 971)	Global unique identifier

_PRL_API_METHOD Data Members

	Name	Description
	MethodAPIVersion (p. 969)	
	MethodIsAsync (p. 969)	
	MethodIsDeprecated (p. 969)	
	MethodIsSrv (p. 969)	
	MethodName (p. 969)	
	MethodPtr (p. 970)	

_PRL_API_METHOD Methods

	Name	Description
	PRL_RESULT (p. 970)	All events (either commands, input/output, etc.) are recieved by the client in this 'callback' functions, provided by the client to the library.

_PRL_API_METHOD Nested Types

Name	Description
PRL_API_METHOD_PTR (p. 970)	
PRL_CONST_API_METHOD_PTR (p. 970)	
PRL_CONST_GUID_PTR (p. 970)	
PRL_FILE_DESC (p. 970)	File descriptor type
PRL_GUID (p. 970)	Global unique identifier
PRL_GUID_PTR (p. 971)	
PRL_RESULT (p. 971)	Unified result error code for each and every method
PRL_RESULT_PTR (p. 971)	

_PRL_API_METHOD Structures

	Name	Description
	_PRL_GUID (p. 971)	Global unique identifier

_PRL_API_METHOD Data Members

_PRL_API_METHOD::MethodAPIVersion Data Member

Syntax

```
PRL_UINT32 MethodAPIVersion;
```

_PRL_API_METHOD::MethodIsAsync Data Member

Syntax

```
PRL_BOOL MethodIsAsync;
```

_PRL_API_METHOD::MethodIsDeprecated Data Member

Syntax

```
PRL_BOOL MethodIsDeprecated;
```

_PRL_API_METHOD::MethodIsSrv Data Member

Syntax

```
PRL_BOOL MethodIsSrv;
```

_PRL_API_METHOD::MethodName Data Member

Syntax

```
PRL_CONST_STR MethodName;
```

_PRL_API_METHOD::MethodPtr Data Member

Syntax

```
PRL_VOID_PTR MethodPtr;
```

_PRL_API_METHOD Methods

_PRL_API_METHOD::PRL_RESULT Method

All events (either commands, input/output, etc.) are recieved by the client in this 'callback' functions, provided by the client to the library.

Syntax

```
PRL_RESULT(
    PRL_CALL * PRL_EVENT_HANDLER_PTR
);
```

_PRL_API_METHOD Nested Types

_PRL_API_METHOD::PRL_API_METHOD_PTR Nested Type

Syntax

```
typedef PRL_API_METHOD* PRL_API_METHOD_PTR;
```

_PRL_API_METHOD::PRL_CONST_API_METHOD_PTR Nested Type

Syntax

```
typedef const PRL_API_METHOD* PRL_CONST_API_METHOD_PTR;
```

_PRL_API_METHOD::PRL_CONST_GUID_PTR Nested Type

Syntax

```
typedef const PRL_GUID* PRL_CONST_GUID_PTR;
```

_PRL_API_METHOD::PRL_FILE_DESC Nested Type

File descriptor type

Syntax

```
typedef PRL_VOID_PTR PRL_FILE_DESC;
```

_PRL_API_METHOD::PRL_GUID Nested Type

Global unique identifier

Syntax

```
typedef struct _PRL_GUID {
  PRL_UINT32 Data1;
  PRL_UINT16 Data2;
  PRL_UINT16 Data3;
  PRL_UINT8 Data4[8];
} PRL_GUID;
```

_PRL_API_METHOD::PRL_GUID_PTR Nested Type

Syntax

```
typedef PRL_GUID* PRL_GUID_PTR;
```

_PRL_API_METHOD::PRL_RESULT Nested Type

Unified result error code for each and every method

Syntax

```
typedef int PRL_RESULT;
```

_PRL_API_METHOD::PRL_RESULT_PTR Nested Type

Syntax

```
typedef PRL_RESULT* PRL_RESULT_PTR;
```

_PRL_API_METHOD Structures

_PRL_API_METHOD::_PRL_GUID Structure

Global unique identifier

Syntax

```
struct _PRL_GUID {
  PRL_UINT32 Data1;
  PRL_UINT16 Data2;
  PRL_UINT16 Data3;
```

```
  PRL_UINT8 Data4[8];
};
```

File

PrlTypes.h

_PRL_APP_IN_DOCK_MODE Enumeration

VM application in Dock mode

Syntax

```
enum _PRL_APP_IN_DOCK_MODE {
  PDM_APP_IN_DOCK_NEVER = 0,
  PDM_APP_IN_DOCK_COHERENCE_ONLY = 1,
  PDM_APP_IN_DOCK_ALWAYS = 2
};
```

File

PrlEnums.h

_PRL_APPLIANCE_DOWNLOAD_STATUS Enumeration

Appliance download status

Syntax

```
enum _PRL_APPLIANCE_DOWNLOAD_STATUS {
  PADS_NOT_STARTED = 1,
  PADS_STOPPED = 2,
  PADS_DOWNLOADING = 3,
  PADS_DOWNLOADED = 4,
  PADS_UNPACKED = 5,
  PADS_MOVED = 6,
  PADS_REGISTERED = 7,
  PADS_CANCELED = 8
};
```

File

PrlEnums.h

_PRL_APPLICATION_MODE Enumeration

Virtuozzo application mode in which SDK should work PAM_SERVER : SDK should working at Virtuozzo Service application mode

Syntax

```
enum _PRL_APPLICATION_MODE {
  PAM_UNKNOWN = 0xffff,
  PAM_SERVER = 0,
  PAM_LAST = PAM_SERVER
};
```

File

PrlEnums.h

Members

PAM_LAST
PAM_LAST should be always last entry in enum !

_PRL_BEGIN_VM_BACKUP_FLAGS Enumeration

PrlVm_BeginBackup flags.

Syntax

```
enum _PRL_BEGIN_VM_BACKUP_FLAGS {
  PBMBF_CREATE_MAP = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Members

PBMBF_CREATE_MAP
Create a disk map

_PRL_BIOS_TYPE Enumeration

BIOS type

Syntax

```
enum _PRL_BIOS_TYPE {
  PBT_LEGACY = 0,
  PBT_EFI = 1
};
```

File

PrlEnums.h

_PRL_CHS Structure

CHS format of disk

Syntax

```
struct _PRL_CHS {
  PRL_UINT32 Sectors;
  PRL_UINT32 Heads;
  PRL_UINT32 Cylinders;
};
```

File

PrlDisk.h

_PRL_CLONE_VM_FLAGS Enumeration

PrlVm_CloneEx (p. 451) command flags set

PCVF_DETACH_EXTERNAL_VIRTUAL_HDD

- to remove link to Virtual Hdd which located outside of VM.pvm directory from cloned vm configuration.

Syntax

```
enum _PRL_CLONE_VM_FLAGS {
  PCVF_CLONE_TO_TEMPLATE = 1<<(PACF_MAX+1),
  PCVF_CHANGE_SID = 1<<(PACF_MAX+2),
  PCVF_LINKED_CLONE = 1<<(PACF_MAX+3),
  PCVF_IMPORT_BOOT_CAMP = 1<<(PACF_MAX+4),
  PCVF_DETACH_EXTERNAL_VIRTUAL_HDD = 1<<(PACF_MAX+5)
};
```

File

PrlCommandsFlags.h

_PRL_CLUSTERED_DEVICE_SUBTYPE Enumeration

Clustered device subtype

Syntax

```
enum _PRL_CLUSTERED_DEVICE_SUBTYPE {
  PCD_BUSLOGIC = 0,
  PCD_LSI_SPI = 1,
  PCD_LSI_SAS = 2,
  PCD_VIRTIO_SCSI = 3,
  PCD_UNKNOWN_DEVICE = 255
};
```

File

PrlEnums.h

_PRL_COMMIT_ENCRYPTION_FLAGS Enumeration

PrlVm_CommitEncryption (p. 454) command flags

Syntax

```
enum _PRL_COMMIT_ENCRYPTION_FLAGS {
  PCEF_CHANGE_MASTER_KEY = 1<<(PACF_MAX+1),
  PCEF_DONT_SHRED_PLAIN_DATA = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

Members

PCEF_CHANGE_MASTER_KEY

Normally, only the master key (MK is used to actually encrypt the data on a virtual disk) is being re-encrypted when the disk encryption key identifier is changed for an already encrypted disk.

PCEF_DONT_SHRED_PLAIN_DATA

When a plain disk is being encrypted, the encrypted copy of the original disk is created.

_PRL_COMPACT_MODE Enumeration

Compact mode: PCM_COMPACT_WITH_HARD_DISKS_INFO : show question before compacting with size disks info PCM_HARD_DISKS_INFO : just get disk info PCM_FULL_CLEAN_UP_VM : compact all disks and delete all unneccessary files

Syntax

```
enum _PRL_COMPACT_MODE {
  PCM_COMPACT_WITH_HARD_DISKS_INFO = 1<<(PACF_MAX+1),
  PCM_HARD_DISKS_INFO = 1<<(PACF_MAX+2),
  PCM_FULL_CLEAN_UP_VM = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_CONVERT_VIRTUAL_DISK Enumeration

Convert virtual disk

- PCVD_CANCEL : cancel disk conversion
- PCVD_TO_PLAIN_DISK : convert disk from expanding to plain disk
- PCVD_TO_EXPANDING_DISK : convert disk from plain to expanding disk
- PCVD_TO_SPLIT_DISK : convert disk from non-split to split disk
- PCVD_TO_NON_SPLIT_DISK : convert disk from split to non-split disk
- PCVD_MERGE_ALL_SNAPSHOTS : merge all disk snapshots

Syntax

```
enum _PRL_CONVERT_VIRTUAL_DISK {
  PCVD_CANCEL = 1<<(PACF_MAX+1),
  PCVD_TO_PLAIN_DISK = 1<<(PACF_MAX+2),
  PCVD_TO_EXPANDING_DISK = 1<<(PACF_MAX+3),
  PCVD_TO_SPLIT_DISK = 1<<(PACF_MAX+4),
  PCVD_TO_NON_SPLIT_DISK = 1<<(PACF_MAX+5),
  PCVD_MERGE_ALL_SNAPSHOTS = 1<<(PACF_MAX+6)
};
```

File

PrlCommandsFlags.h

_PRL_COPY_CT_TEMPLATE_FLAGS Enumeration

copy of CT templates functionality flags set

Syntax

```
enum _PRL_COPY_CT_TEMPLATE_FLAGS {
  PCTMP_LOW_SECURITY = PSL_LOW_SECURITY,
  PCTMP_NORMAL_SECURITY = PACF_NORMAL_SECURITY,
  PCTMP_HIGH_SECURITY = PACF_HIGH_SECURITY,
  PCTMPL_FORCE = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Type Parameters

PCTMPL_FORCE

force template copy

_PRL_CORNER_ACTIONS Enumeration

Window corner actions

Syntax

```
enum _PRL_CORNER_ACTIONS {
  PCA_NO_ACTION = 0,
  PCA_WINDOWED = 1,
  PCA_COHERENCE = 2,
  PCA_SEAMLESS = 3,
  PCA_MODALITY = 4,
  PCA_HIDE_APPLICATION = 5,
  PCA_UNGRAB_INPUT = 6,
  PCA_CRYSTAL = 7
};
```

File

PrlEnums.h

_PRL_CPU_FEATURES_EX Enumeration

Syntax

```
enum _PRL_CPU_FEATURES_EX {
  PCFE_FEATURES = 0,
  PCFE_EXT_FEATURES = 1,
  PCFE_EXT_00000007_EBX = 2,
  PCFE_EXT_80000001_ECX = 3,
  PCFE_EXT_80000001_EDX = 4,
  PCFE_EXT_80000007_EDX = 5,
  PCFE_EXT_80000008_EAX = 6,
```

```
  PCFE_EXT_0000000D_EAX = 7,
  PCFE_MAX
};
```

File

PrlEnums.h

Members

PCFE_MAX
have to be the last.

_PRL_CPU_HVT Enumeration

CPU mode

Syntax

```
enum _PRL_CPU_HVT {
  PCM_CPU_NONE_HV = 0,
  PCM_CPU_INTEL_VT_X,
  PCM_CPU_AMD_V
};
```

File

PrlEnums.h

_PRL_CPU_MODE Enumeration

CPU mode

Syntax

```
enum _PRL_CPU_MODE {
  PCM_CPU_MODE_32 = 0,
  PCM_CPU_MODE_64 = 1
};
```

File

PrlEnums.h

_PRL_CPULIMIT_TYPE Enumeration

Syntax

```
enum _PRL_CPULIMIT_TYPE {
  PRL_CPULIMIT_PERCENTS100 = 1,
  PRL_CPULIMIT_MHZ = 2,
  PRL_CPULIMIT_PERCENTS = 3,
  PRL_CPULIMIT_PERCENTS_TO_MHZ = 4
};
```

File

PrlEnums.h

Members

PRL_CPULIMIT_PERCENTS100
cpulimit 0-100

PRL_CPULIMIT_PERCENTS
cpulimit 0-100 * nCpu

PRL_CPULIMIT_PERCENTS_TO_MHZ
The limit in % convert to Mhz

_PRL_CPUS_COUNT Enumeration

constant for unlimited CPU number

Syntax

```
enum _PRL_CPUS_COUNT {
  PRL_CPU_UNLIMITED = 0xffffffff
};
```

File

PrlEnums.h

_PRL_CPUUNITS Enumeration

Syntax

```
enum _PRL_CPUUNITS {
  PRL_CPUUNITS_MIN = 10,
  PRL_CPUUNITS_DEFAULT = 1000,
  PRL_CPUUNITS_MAX = 500000
};
```

File

PrlEnums.h

_PRL_CREATE_DISK_IMAGE_FLAGS Enumeration

Create image flags

Syntax

```
enum _PRL_CREATE_DISK_IMAGE_FLAGS {
  PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Members

PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION
create from lion recovery

_PRL_CREATE_SNAPSHOT_FLAGS Enumeration

VM snapshot creation flags set

Syntax

```
enum _PRL_CREATE_SNAPSHOT_FLAGS {
  PCSF_DISK_ONLY = 1<<(PACF_MAX+1),
  PCSF_BACKUP = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

Members

PCSF_DISK_ONLY
internal flag

PCSF_BACKUP
internal flag

Type Parameters

PCSF_DISK_ONLY
to snapshot hdd's only for running Vm

PCSF_BACKUP
create snapshot for backup

_PRL_CT_CAPMASK Enumeration

Capabilities list

Syntax

```
enum _PRL_CT_CAPMASK {
  PCC_CHOWN = (1 << 0),
  PCC_FS_MASK = PCC_CHOWN,
  PCC_DAC_OVERRIDE = (1 << 1),
  PCC_DAC_READ_SEARCH = (1 << 2),
  PCC_FOWNER = (1 << 3),
  PCC_FSETID = (1 << 4),
  PCC_KILL = (1 << 5),
  PCC_SETGID = (1 << 6),
  PCC_SETUID = (1 << 7),
  PCC_SETPCAP = (1 << 8),
  PCC_LINUX_IMMUTABLE = (1 << 9),
  PCC_NET_BIND_SERVICE = (1 << 10),
  PCC_NET_BROADCAST = (1 << 11),
  PCC_NET_ADMIN = (1 << 12),
  PCC_NET_RAW = (1 << 13),
```

```
    PCC_IPC_LOCK = (1 << 14),
    PCC_IPC_OWNER = (1 << 15),
    PCC_SYS_MODULE = (1 << 16),
    PCC_SYS_RAWIO = (1 << 17),
    PCC_SYS_CHROOT = (1 << 18),
    PCC_SYS_PTRACE = (1 << 19),
    PCC_SYS_PACCT = (1 << 20),
    PCC_SYS_ADMIN = (1 << 21),
    PCC_SYS_BOOT = (1 << 22),
    PCC_SYS_NICE = (1 << 23),
    PCC_SYS_RESOURCE = (1 << 24),
    PCC_SYS_TIME = (1 << 25),
    PCC_SYS_TTY_CONFIG = (1 << 26),
    PCC_MKNOD = (1 << 27),
    PCC_LEASE = (1 << 28),
    PCC_AUDIT_WRITE = (1 << 29),
    PCC_VE_ADMIN = (1 << 30),
    PCC_SETFCAP = (1 << 31)
};
```

File

PrlEnums.h

Members

PCC_FS_MASK
FIXME: real value of PCC_FS_MASK?

_PRL_CT_FEATURES Enumeration

CT Features

Syntax

```
enum _PRL_CT_FEATURES {
    PCF_FEATURE_SYSFS = (1 << 0),
    PCF_FEATURE_NFS = (1 << 1),
    PCF_FEATURE_DEF_PERM = (1 << 2),
    PCF_FEATURE_SIT = (1 << 3),
    PCF_FEATURE_IPIP = (1 << 4),
    PCF_FEATURE_PPP = (1 << 5),
    PCF_FEATURE_IPGRE = (1 << 6),
    PCF_FEATURE_BRIDGE = (1 << 7),
    PCF_FEATURE_NFSD = (1 << 8)
};
```

File

PrlEnums.h

_PRL_CT_REINSTALL_OPTIONS Enumeration

Syntax

```
enum _PRL_CT_REINSTALL_OPTIONS {
    REINSTALL_SKIP_BACKUP = 1<<(PACF_MAX+1),
    REINSTALL_SKIP_SCRIPTS = 2<<(PACF_MAX+1),
    REINSTALL_RESET_PWDB = 3<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

_PRL_CT_RESOURCE Enumeration

Syntax

```
enum _PRL_CT_RESOURCE {
  PCR_KMEMSIZE = 1,
  PCR_LOCKEDPAGES = 2,
  PCR_PRIVVMPAGES = 3,
  PCR_SHMPAGES = 4,
  PCR_NUMPROC = 5,
  PCR_PHYSPAGES = 6,
  PCR_VMGUARPAGES = 7,
  PCR_OOMGUARPAGES = 8,
  PCR_NUMTCPSOCK = 9,
  PCR_NUMFLOCK = 10,
  PCR_NUMPTY = 11,
  PCR_NUMSIGINFO = 12,
  PCR_TCPSNDBUF = 13,
  PCR_TCPRCVBUF = 14,
  PCR_OTHERSOCKBUF = 15,
  PCR_DGRAMRCVBUF = 16,
  PCR_NUMOTHERSOCK = 17,
  PCR_DCACHESIZE = 18,
  PCR_NUMFILE = 19,
  PCR_NUMIPTENT = 20,
  PCR_SWAPPAGES = 21,
  PCR_QUOTAUGIDLIMIT = 22,
  PCR_LAST = PCR_QUOTAUGIDLIMIT
};
```

File

PrlEnums.h

_PRL_CT_TEMPLATE_TYPE Enumeration

Container templates types

Syntax

```
enum _PRL_CT_TEMPLATE_TYPE {
  PCT_TYPE_EZ_OS = 0,
  PCT_TYPE_EZ_APP = 1
};
```

File

PrlEnums.h

_PRL_DATA_STATISTIC_SEGMENTS Enumeration

Data statistic segments

Syntax

```
enum _PRL_DATA_STATISTIC_SEGMENTS {
  PDSS_UNKNOWN = 0,
  PDSS_VM_MISCELLANEOUS_SPACE = 1,
  PDSS_VM_RECLAIM_SPACE = 2,
  PDSS_VM_SNAPSHOTS_SPACE = 3,
  PDSS_VM_DISK_DATA_SPACE = 4,
  PDSS_VM_FULL_SPACE = 5
};
```

File

PrlEnums.h

_PRL_DELETE_SNAPSHOT_FLAGS Enumeration

- PrlVm_DeleteSnapshotEx flags PDSF_BACKUP - delete snapshot for backup

Syntax

```
enum _PRL_DELETE_SNAPSHOT_FLAGS {
  PDSF_SKIP_SPACE_CHECK = 1<<(PACF_MAX+1),
  PDSF_BACKUP = 1<<(PACF_MAX+2),
  PDSF_BACKUP_MAP = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

Members

PDSF_SKIP_SPACE_CHECK
skip space check

PDSF_BACKUP
internal flag

PDSF_BACKUP_MAP
internal flag

_PRL_DEVICE_ACTION Enumeration

Device in use Action

Syntax

```
enum _PRL_DEVICE_ACTION {
  PDA_DEVICE_IDLE = 0,
  PDA_DEVICE_READ = 1,
  PDA_DEVICE_WRITE = 2
```

```
};
```

File

PrlEnums.h

_PRL_DEVICE_TYPE Enumeration

Device type

Syntax

```
enum _PRL_DEVICE_TYPE {
  PDE_GENERIC_DEVICE = 0,
  PDE_CLUSTERED_DEVICE = 1,
  PDE_STORAGE_DEVICE = 2,
  PDE_FLOPPY_DISK = 3,
  PDE_MASSSTORAGE_DEVICE = 4,
  PDE_OPTICAL_DISK = 5,
  PDE_HARD_DISK = 6,
  PDE_GENERIC_NETWORK_ADAPTER = 8,
  PDE_GENERIC_PORT = 9,
  PDE_SERIAL_PORT = 10,
  PDE_PARALLEL_PORT = 11,
  PDE_SOUND_DEVICE = 12,
  PDE_MIXER_DEVICE = 13,
  PDE_USB_DEVICE = 15,
  PDE_PRINTER = 16,
  PDE_GENERIC_PCI_DEVICE = 17,
  PDE_GENERIC_SCSI_DEVICE = 18,
  PDE_VIRTUAL_SNAPSHOT_DEVICE = 19,
  PDE_PCI_VIDEO_ADAPTER = 20,
  PDE_VIRTUAL_SHARED_FOLDERS_DEVICE = 21,
  PDE_ATTACHED_BACKUP_DISK = 22,
  PDE_MAX
};
```

File

PrlEnums.h

_PRL_DISPLAY_BUFFER_FORMAT Enumeration

Format of display buffer

Syntax

```
enum _PRL_DISPLAY_BUFFER_FORMAT {
  PDBF_NO_CONVERSION = 0,
  PDBF_INDEX8 = 1,
  PDBF_RGB15 = 2,
  PDBF_RGB16 = 3,
  PDBF_RGB24 = 4,
  PDBF_BGR32 = 5,
  PDBF_RGB32 = 6,
  PDBF_YUV422x8 = 10,
  PDBF_YUV422x6 = 11,
  PDBF_YUV422x4 = 12,
  PDBF_YUV420 = 16
};
```

File

PrlEnums.h

_PRL_DISPLAY_CODEC_COMPRESSION Enumeration

Obsolete enum.

Syntax

```
enum _PRL_DISPLAY_CODEC_COMPRESSION {
  PDCC_HIGH_COMPRESSION = 1<<15,
  PDCC_MEDIUM_COMPRESSION = 1<<16,
  PDCC_LOW_COMPRESSION = 1<<17,
  PDCC_NO_COMPRESSION = 1<<18
};
```

File

PrlEnums.h

Remarks

Use display encodings at runtime for connection with new server

Notes

#PrlSrv_GetDisplayEncodingList Display codec compression

_PRL_DISPLAY_CODEC_QUALITY Enumeration

Obsolete enum.

Syntax

```
enum _PRL_DISPLAY_CODEC_QUALITY {
  PDCQ_HIGH_QUALITY = 1<<0,
  PDCQ_MEDIUM_QUALITY = 1<<1,
  PDCQ_LOW_QUALITY = 1<<2
};
```

File

PrlEnums.h

Remarks

Use display encodings at runtime for connection with new server

Notes

#PrlSrv_GetDisplayEncodingList Display codec quality.

_PRL_DISPLAY_CODEC_TYPE Enumeration

Obsolete enum.

Syntax

```
enum _PRL_DISPLAY_CODEC_TYPE {
  PDCT_HIGH_QUALITY_WITHOUT_COMPRESSION = PDCQ_HIGH_QUALITY|PDCC_NO_COMPRESSION,
  PDCT_HIGH_QUALITY_WITH_COMPRESSION = PDCQ_HIGH_QUALITY|PDCC_MEDIUM_COMPRESSION,
  PDCT_MEDIUM_QUALITY_WITHOUT_COMPRESSION = PDCQ_MEDIUM_QUALITY|PDCC_NO_COMPRESSION,
  PDCT_MEDIUM_QUALITY_WITH_COMPRESSION = PDCQ_MEDIUM_QUALITY|PDCC_MEDIUM_COMPRESSION,
  PDCT_LOW_QUALITY_WITHOUT_COMPRESSION = PDCQ_LOW_QUALITY|PDCC_NO_COMPRESSION,
  PDCT_LOW_QUALITY_WITH_COMPRESSION = PDCQ_LOW_QUALITY|PDCC_MEDIUM_COMPRESSION
};
```

File

PrlEnums.h

Remarks

Use display encodings at runtime for connection with new server

Notes

#PrlSrv_GetDisplayEncodingList Display codec types

_PRL_DISPLAY_CONFIG_MODE Enumeration

Display(s) configuration mode

Syntax

```
enum _PRL_DISPLAY_CONFIG_MODE {
  PDCM_CLIENT_DISPLAY_INFO = (1 << 0),
  PDCM_HAS_HIDPI_DISPLAY = (1 << 1),
  PDCM_DO_NOT_RESTORE = (1 << 2),
  PDCM_RESTORE_DEFAULT = (1 << 3),
  PDCM_DISABLE_HIDPI_ADJUSTMENT = (1 << 4),
  PDCM_RETINA_MODE = (1 << 5),
  PDCM_ENABLE_HEADS = (1 << 6)
};
```

File

PrlEnums.h

_PRL_DISTRO_MEDIA_TYPE Enumeration

Syntax

```
enum _PRL_DISTRO_MEDIA_TYPE {
  DMT_DISTRO = 0,
  DMT_LIVE_CD,
  DMT_MBR
};
```

File

PrlEnums.h

Members

DMT_DISTRO

Ordinary distro

DMT_LIVE_CD
LiveCD

DMT_MBR
Media has Master Boot Record

_PRL_DOCK_ICON_TYPE Enumeration

VM application icon type in Dock

Syntax

```
enum _PRL_DOCK_ICON_TYPE {
  PIT_DOCK_ICON_SYSTEM = 0,
  PIT_DOCK_ICON_LIVE_SCREEN_SHOT = 1,
  PIT_DOCK_ICON_START_MENU = 2
};
```

File

PrlEnums.h

_PRL_END_VM_BACKUP_FLAGS Enumeration

PrlVmBackup_Commit / PrlVmBackup_Rollback flags.

Syntax

```
enum _PRL_END_VM_BACKUP_FLAGS {
  PEMBF_FAILURE_CLEANUP = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Members

PEMBF_FAILURE_CLEANUP
Perform failure specific clean-up

_PRL_EVENT_ISSUER_TYPE Enumeration

Event issuer type

Syntax

```
enum _PRL_EVENT_ISSUER_TYPE {
  PIE_VIRTUAL_MACHINE = 0,
  PIE_DISPATCHER = 1,
  PIE_IO_SERVICE = 2,
  PIE_WEB_SERVICE = 3,
  PIE_UNKNOWN = 0xFF
};
```

File

PrlEnums.h

_PRL_EVENT_TYPE Enumeration

Different event types are identified by these event codes.

Syntax

```
enum _PRL_EVENT_TYPE {
  PET_VM_INF_UNINITIALIZED_EVENT_CODE = 100000,
  PET_DSP_EVT_VM_STATE_CHANGED = 100001,
  PET_DSP_EVT_VM_CONFIG_CHANGED = 100002,
  PET_DSP_EVT_VM_DEV_STATE_CHANGED = 100003,
  PET_DSP_EVT_VM_CREATED = 100004,
  PET_DSP_EVT_VM_ADDED = 100005,
  PET_DSP_EVT_VM_DELETED = 100006,
  PET_DSP_EVT_VM_STARTED = 100007,
  PET_DSP_EVT_VM_STOPPED = 100008,
  PET_DSP_EVT_VM_ABORTED = 100009,
  PET_DSP_EVT_VM_RESETED = 100010,
  PET_DSP_EVT_VM_SUSPENDING = 100011,
  PET_DSP_EVT_VM_SUSPENDED = 100012,
  PET_DSP_EVT_VM_RESUMING = 100013,
  PET_DSP_EVT_VM_RESUMED = 100014,
  PET_DSP_EVT_VM_START_IO_FLOW = 100015,
  PET_DSP_EVT_VM_SNAPSHOTED = 100016,
  PET_DSP_EVT_VM_STATESAVED = PET_DSP_EVT_VM_SNAPSHOTED,
  PET_DSP_EVT_VM_RESTORED = 100017,
  PET_DSP_EVT_VM_STATERESTORED = PET_DSP_EVT_VM_RESTORED,
  PET_DSP_EVT_VM_NET_RECONNECTED = 100018,
  PET_DSP_EVT_VM_PAUSED = 100019,
  PET_DSP_EVT_VM_CONTINUED = 100020,
  PET_DSP_EVT_VM_CONFIG_APPLIED = 100021,
  PET_DSP_EVT_VM_STATISTICS_UPDATED = 100022,
  PET_DSP_EVT_VM_PROBLEM_REPORT_CREATED = 100023,
  PET_DSP_EVT_VM_ABOUT_TO_RESET = 100024,
  PET_DSP_EVT_VM_SECURITY_CHANGED = 100025,
  PET_DSP_EVT_VM_UNREGISTERED = 100026,
  PET_DSP_EVT_VM_REMOTE_DEV_READY_TO_START = 100027,
  PET_VM_EVT_VM_CONFIG_CHANGED = 100028,
  PET_VM_EVT_REQUEST_FILE_OPEN = 100029,
  PET_DSP_EVT_VM_MIGRATE_STARTED = 100030,
  PET_DSP_EVT_VM_MIGRATE_CANCELLED = 100031,
  PET_DSP_EVT_VM_MIGRATE_FINISHED = 100032,
  PET_DSP_EVT_VM_MIGRATE_PROGRESS_CHANGED = 100033,
  PET_DSP_EVT_VM_SNAPSHOTING = 100034,
  PET_DSP_EVT_VM_RESTORING = 100035,
  PET_DSP_EVT_VM_SNAPSHOTS_TREE_CHANGED = 100036,
  PET_DSP_EVT_VM_DELETING_STATE = 100037,
  PET_DSP_EVT_VM_STATE_DELETED = 100038,
  PET_DSP_EVT_VM_ABOUT_TO_START_DEINIT = 100039,
  PET_DSP_EVT_VM_ABOUT_TO_START_INIT = 100040,
  PET_DSP_EVT_VM_MEMORY_SWAPPING_STARTED = 100041,
  PET_DSP_EVT_VM_MEMORY_SWAPPING_FINISHED = 100042,
  PET_VM_EVT_UNMOUNT = 100043,
  PET_VM_EVT_MOUNT = 100044,
  PET_DSP_EVT_REBOOT_HOST = 100045,
  PET_DSP_EVT_REBOOT_HOST_MANUAL_TO_INIT_VTD_DEVICES = 100046,
  PET_DSP_EVT_BACKUP_STARTED = 100047,
  PET_DSP_EVT_BACKUP_CANCELLED = 100048,
```

```
    PET_DSP_EVT_CREATE_BACKUP_FINISHED = 100049,
    PET_DSP_EVT_BACKUP_PROGRESS_CHANGED = 100050,
    PET_DSP_EVT_JOB_PROGRESS_CHANGED = 100051,
    PET_DSP_EVT_RESTORE_PROGRESS_CHANGED = 100052,
    PET_DSP_EVT_DISK_RESIZE_STARTED = 100053,
    PET_DSP_EVT_DISK_RESIZE_FINISHED = 100054,
    PET_DSP_EVT_DISK_RESIZE_PROGRESS_CHANGED = 100055,
    PET_DSP_EVT_VM_ADDITION_STATE_CHANGED = 100056,
    PET_DSP_EVT_RESTORE_BACKUP_FINISHED = 100057,
    PET_DSP_EVT_REMOVE_BACKUP_FINISHED = 100058,
    PET_DSP_EVT_VM_MIGRATE_CANCELLED_DISP = 100059,
    PET_DSP_EVT_VM_MIGRATE_FINISHED_DISP = 100060,
    PET_DSP_EVT_TIME_MACHINE_BACKUP_IS_OVER = 100061,
    PET_DSP_EVT_CONVERSION_DISKS_PROGRESS_CHANGED = 100062,
    PET_DSP_EVT_CONVERSION_DISKS_PROGRESS_FINISHED = 100063,
    PET_DSP_EVT_CONVERT_THIRD_PARTY_PROGRESS_CHANGED = 100064,
    PET_DSP_EVT_CONNECTED_TO_PROXY = 100065,
    PET_DSP_EVT_DISCONNECTED_FROM_PROXY = 100066,
    PET_DSP_EVT_CONNECTING_TO_PROXY = 100067,
    PET_DSP_EVT_RECONFIG_VM_PROGRESS_CHANGED = 100068,
    PET_DSP_EVT_VM_MIGRATE_WAIT_REMOUNT = 100069,
    PET_DSP_EVT_VM_CONFIG_APPLY_FINISHED = 100070,
    PET_DSP_EVT_VM_IO_CLIENT_STATISTICTS = 100071,
    PET_DSP_EVT_VM_SET_HOST_TIME = 100072,
    PET_DSP_EVT_VM_LOW_HOST_MEMORY = 100073,
    PET_DSP_EVT_VM_MAC_OS_VER = 100074,
    PET_DSP_EVT_VM_PAUSED_BY_HOST_SLEEP = 100075,
    PET_DSP_EVT_VM_CONTINUED_BY_HOST_WAKEUP = 100076,
    PET_DSP_EVT_VM_FROZEN = 100077,
    PET_DSP_EVT_VM_UNFROZEN = 100078,
    PET_DSP_EVT_CONNECTION_CLIENT_INFO_WAS_SET = 100079,
    PET_DSP_EVT_CONNECTION_WAS_CLOSED = 100080,
    PET_DSP_EVT_OP_ACCEPTED = 100100,
    PET_DSP_EVT_OP_REJECTED = 100101,
    PET_DSP_EVT_OP_RESULT = 100102,
    PET_DSP_EVT_DIR_STATE_CHANGED = 100200,
    PET_DSP_EVT_HOST_STATISTICS_UPDATED = 100201,
    PET_DSP_EVT_HW_CONFIG_CHANGED = 100202,
    PET_DSP_EVT_CLIENT_CONNECTED = 100300,
    PET_DSP_EVT_CLIENT_DISCONNECTED = 100301,
    PET_DSP_EVT_CLIENT_REJECTED = 100302,
    PET_DSP_EVT_FOUND_LOST_VM_CONFIG = 100400,
    PET_DSP_EVT_HAS_EVENT = 100401,
    PET_DSP_EVT_RECONNECT = 100402,
    PET_DSP_EVT_DISP_CONNECTION_CLOSED = 100403,
    PET_DSP_EVT_DISP_SHUTDOWN = 100404,
    PET_DSP_EVT_USER_PROFILE_CHANGED = 100500,
    PET_DSP_EVT_COMMON_PREFS_CHANGED = 100501,
    PET_DSP_EVT_VM_MESSAGE = 100502,
    PET_DSP_EVT_VM_QUESTION = 100503,
    PET_JOB_DELETE_VM_PROGRESS_CHANGED = 100504,
    PET_VM_INF_START_FILE_COPYING = 100505,
    PET_VM_INF_END_FILE_COPYING = 100506,
    PET_DSP_EVT_ERROR_MESSAGE = 100507,
    PET_DSP_EVT_WARNING_MESSAGE = 100508,
    PET_JOB_HDD_CREATE_PROGRESS_CHANGED = 100509,
    PET_JOB_FILE_COPY_PROGRESS_CHANGED = 100510,
    PET_JOB_SUSPEND_PROGRESS_CHANGED = 100511,
    PET_JOB_RESUME_PROGRESS_CHANGED = 100512,
    PET_DSP_EVT_ANSWER_TO_VM_WAS_DONE = 100513,
    PET_DSP_EVT_LICENSE_CHANGED = 100514,
    PET_JOB_CREATE_SNAPSHOT_PROGRESS_CHANGED = 100515,
    PET_JOB_SWITCH_TO_SNAPSHOT_PROGRESS_CHANGED = 100516,
    PET_JOB_DELETE_SNAPSHOT_PROGRESS_CHANGED = 100517,
    PET_JOB_COMMIT_UNFINISHED_DISK_OP_PROGRESS_CHANGED = 100518,
    PET_DSP_EVT_COMMIT_UNFINISHED_DISK_OP_STARTED = 100519,
```

```
    PET_DSP_EVT_COMMIT_UNFINISHED_DISK_OP_FINISHED = 100520,
    PET_DSP_EVT_VM_ENCRYPT_PROGRESS_CHANGED = 100521,
    PET_DSP_EVT_VM_DECRYPT_PROGRESS_CHANGED = 100522,
    PET_JOB_STAGE_PROGRESS_CHANGED = 100523,
    PET_VM_INF_START_BUNCH_COPYING = 100524,
    PET_VM_INF_END_BUNCH_COPYING = 100525,
    PET_DSP_EVT_SMC_USER_FORCE_DISCONNECTED = 100600,
    PET_DSP_EVT_SMC_CANCEL_USER_COMMAND = 100601,
    PET_DSP_EVT_SMC_DISPATCHER_SHUTDOWN = 100602,
    PET_IO_SCREEN_SIZE = 100700,
    PET_IO_SLIDING_MOUSE_FLAG = 100702,
    PET_IO_DEVICE_IS_IN_USE = 100703,
    PET_IO_KEYBOARD_LEDS = 100704,
    PET_IO_STATE = 100705,
    PET_IO_SCREEN_CAPTURED = 100706,
    PET_IO_BEFORE_SCREEN_SIZE = 100707,
    PET_IO_REMOTE_COMMAND = 100708,
    PET_IO_MOUSE_CURSOR_CHANGED = 100709,
    PET_IO_MOUSE_CURSOR_HID = 100710,
    PET_IO_MOUSE_CURSOR_SET = 100711,
    PET_IO_DYNRES_TOOL_STATUS = 100712,
    PET_IO_TOOLS_UTILITY_COMMAND = 100713,
    PET_IO_TOOLS_VM_SHUTDOWN = 100714,
    PET_IO_TOOLS_CLIPBOARD_DATA = 100715,
    PET_IO_SCREEN_SURFACE_DETACHED = 100716,
    PET_IO_TOOLS_SIA_DATA = 100717,
    PET_IO_TOOLS_VMCTG_COMMAND = 100718,
    PET_IO_TOOLS_DRAGDROP_DATA = 100719,
    PET_IO_TOOLS_GENERAL_COMMAND = 100720,
    PET_IO_TOOLS_DESKTOP_UTILITY_STATE = 100721,
    PET_IO_DISPLAY_SCREEN_SIZE = 100722,
    PET_IO_DISPLAY_BEFORE_SCREEN_SIZE = 100723,
    PET_IO_DISPLAY_SCREEN_CAPTURED = 100724,
    PET_IO_SWITCH_VESA_MODE = 100725,
    PET_IO_TOOLS_LANGUAGE_HOTKEY_CHANGED = 100726,
    PET_IO_AVAILABLE_DISPLAYS = 100727,
    PET_IO_SCREEN_SURFACE_ATTACHED = 100728,
    PET_IO_UIEMU_ELEMENT_AT_POS = 100729,
    PET_IO_AUDIO_OUTPUT_DATA = 100730,
    PET_IO_AUDIO_OUTPUT_ENCODING_SET = 100731,
    PET_IO_AUDIO_OUTPUT_STREAM_STARTED = 100732,
    PET_IO_AUDIO_OUTPUT_STOP = 100734,
    PET_IO_CVSRC_OPEN = 100735,
    PET_IO_CVSRC_CLOSE = 100736,
    PET_IO_SSO_GET_CREDENTIALS = 100737,
    PET_IO_MOUSE_CURSOR_MOVED = 100738,
    PET_IO_UIEMU_CARET_INFO = 100739,
    PET_IO_DISPLAY_GAMMA_CHANGED = 100740,
    PET_DSP_EVT_VM_TOOLS_STATE_CHANGED = 100800,
    PET_DSP_EVT_VM_COMPACT_PROCESSING = 100801,
    PET_DSP_EVT_VM_COMPACT_FINISHED = 100802,
    PET_DSP_EVT_VM_COMPRESSOR_FINISHED = 100803,
    PET_DSP_EVT_VM_COMPRESSOR_CONTINUED = 100804,
    PET_DSP_EVT_VM_SOFTWARE_INSTALLED = 100805,
    PET_DSP_EVT_VM_PIS_NOTIFICATION_ASK_TO_INSTALL = 100806,
    PET_DSP_EVT_VM_VIRTUAL_DEVICES_STATE_CHANGED = 100807,
    PET_DSP_EVT_PERFSTATS = 100900,
    PET_DSP_EVT_VM_PERFSTATS = 100901,
    PET_DSP_EVT_VM_UPGRADE_INIT = 101000,
    PET_DSP_EVT_VM_UPGRADE_INIT_TIMEOUT = 101001,
    PET_DSP_EVT_VM_UPGRADE_STAGE_1 = 101002,
    PET_DSP_EVT_VM_UPGRADE_STAGE_2 = 101003,
    PET_DSP_EVT_VM_UPGRADE_STAGE_3 = 101004,
    PET_DSP_EVT_VM_UPGRADE_COMPLETED = 101005,
    PET_DSP_EVT_VM_UPGRADE_UNKNOWN_ERROR = 101006,
    PET_DSP_EVT_VM_UPGRADE_INIT_VMW = 101007,
```

```
    PET_DSP_EVT_VM_UPGRADE_INIT_PHY = 101008,
    PET_IO_READY_TO_ACCEPT_STDIN_PKGS = 101100,
    PET_VM_READY_TO_BOOT = 101200,
    PET_VM_READY_TO_PRINT = 101201,
    PET_VM_DEFAULT_SOUND_RECONNECTED = 101202,
    PET_VM_EVT_UNMOUNT_USB_MEDIA = 101203,
    PET_VM_EVT_UNMOUNT_DVD_MEDIA = 101204,
    PET_DSP_EVT_APPLIANCE_DOWNLOAD_PROGRESS_CHANGED = 101300,
    PET_DSP_EVT_APPLIANCE_DOWNLOAD_FINISHED = 101301,
    PET_DSP_EVT_APPLIANCE_ARCHIVE_UNPACK_STARTED = 101302,
    PET_DSP_EVT_APPLIANCE_ARCHIVE_UNPACK_FINISHED = 101303,
    PET_DSP_EVT_APPLIANCE_REGISTER_VM_STARTED = 101304,
    PET_DSP_EVT_APPLIANCE_REGISTER_VM_FINISHED = 101305,
    PET_DSP_EVT_CEP_WRITE_STRING = 101400,
    PET_DSP_COHERENCE_FAKE_STUB_NOTIFICATION = 101401,
    PET_DSP_EVT_VM_WAS_ENCRYPTED = 101500,
    PET_DSP_EVT_VM_WAS_DECRYPTED = 101501,
    PET_DSP_EVT_ENCRYPTED_VM_PASSWORD_CHANGED = 101502,
    PET_DSP_EVT_ENCRYPTED_VM_NEED_AUTH_AGAIN = 101503,
    PET_DSP_EVT_VM_WAS_PROTECTED = 101504,
    PET_DSP_EVT_VM_WAS_UNPROTECTED = 101505,
    PET_VM_EVT_TIS_BACKUP_WRITE = 101600,
    PET_IO_EVT_DISCONNECT_WITH_REASON = 101700,
    PET_VIDEO_RECEIVER_FORMAT = 101800,
    PET_VIDEO_RECEIVER_FRAME = 101801,
    PET_VIDEO_RECEIVER_CLOSED = 101802,
    PET_VIDEO_CAPTURE_GEOMETRY = 101803,
    PET_VIDEO_RECEIVER_CONNECTION_STATE = 101804,
    PET_PTM_EVT_PROPERTY_CHANGED = 101900,
    PET_PTM_EVT_OBJECT_DELETED = 101901,
    PET_PTM_EVT_OBJECT_CREATED = 101902,
    PET_PTM_EVT_SUBSCRIBED = 101903,
    PET_PTM_EVT_RPC_COMPLETED = 101904
};
```

File

PrlEventsValues.h

Members

PET_VM_INF_UNINITIALIZED_EVENT_CODE
BASE VALUE SET TO 100000

PET_DSP_EVT_OP_ACCEPTED
ALIGNED VALUE TO 100100

PET_DSP_EVT_DIR_STATE_CHANGED
ALIGNED VALUE TO 100200

PET_DSP_EVT_CLIENT_CONNECTED
ALIGNED VALUE TO 100300

PET_DSP_EVT_FOUND_LOST_VM_CONFIG
ALIGNED VALUE TO 100400

PET_DSP_EVT_USER_PROFILE_CHANGED
ALIGNED VALUE TO 100500

PET_DSP_EVT_SMC_USER_FORCE_DISCONNECTED
ALIGNED VALUE TO 100600

PET_IO_SCREEN_SIZE
ALIGNED VALUE TO 100700

PET_DSP_EVT_VM_TOOLS_STATE_CHANGED

ALIGNED VALUE TO 100800

PET_DSP_EVT_PERFSTATS
ALIGNED VALUE TO 100900

PET_DSP_EVT_VM_UPGRADE_INIT
ALIGNED VALUE TO 101000

PET_IO_READY_TO_ACCEPT_STDIN_PKGS
ALIGNED VALUE TO 101100

PET_VM_READY_TO_BOOT
ALIGNED VALUE TO 101200

PET_DSP_EVT_APPLIANCE_DOWNLOAD_PROGRESS_CHANGED
ALIGNED VALUE TO 101300

PET_DSP_EVT_CEP_WRITE_STRING
ALIGNED VALUE TO 101400

PET_DSP_EVT_VM_WAS_ENCRYPTED
ALIGNED VALUE TO 101500

PET_VM_EVT_TIS_BACKUP_WRITE
ALIGNED VALUE TO 101600

PET_IO_EVT_DISCONNECT_WITH_REASON
ALIGNED VALUE TO 101700

PET_VIDEO_RECEIVER_FORMAT
ALIGNED VALUE TO 101800

PET_PTM_EVT_PROPERTY_CHANGED
ALIGNED VALUE TO 101900

Remarks

Client can query event handle with the PrlEvent_GetType (p. 86) call to get this code. @see PrlEvent_GetType (p. 86)

_PRL_EXTERNAL_VM_TYPE Enumeration

External VM type for supporting native VM

Syntax

```
enum _PRL_EXTERNAL_VM_TYPE {
  PEVT_VM_UNKNOWN = 0,
  PEVT_NATIVE_VM = PEVT_VM_UNKNOWN,
  PEVT_VMWARE = 1
};
```

File

PrlEnums.h

_PRL_FEATURES_MATRIX Enumeration

Features matrix list

Syntax

```
enum _PRL_FEATURES_MATRIX {
  PFSM_UNKNOWN_FEATURE = 0,
  PFSM_SATA_HOTPLUG_SUPPORT = 1,
  PFSM_DEFAULT_SATA_ALLOWED = PFSM_SATA_HOTPLUG_SUPPORT,
  PFSM_VM_CONFIG_MERGE_SUPPORT = 2,
  PFSM_PSBM5 = 3,
  PFSM_AUTOSTART_VM_AS_OWNER = 4,
  PFSM_DEFAULT_PLAINDISK_ALLOWED = 5,
  PFSM_NO_SHARED_NETWORKING = 6,
  PFSM_DISK_IO_LIMITS = 7,
  PFSM_FINE_CPU_LIMITS = 8,
  PFSM_ROUTED_NETWORKING = 9,
  PFSM_RAM_HOTPLUG_SUPPORT = 10,
  PFSM_CPU_HOTPLUG_SUPPORT = 11,
  PFSM_IPV6_SUPPORT = 12,
  PFSM_USB_PRINTER_SUPPORT = 13,
  PFSM_NIC_CHANGE_ALLOWED = 14,
  PFSM_DESKTOP_CONTROL_SUPPORT = 15
};
```

File

PrlEnums.h

_PRL_FILE_SYSTEM_ELEMENT_TYPE Enumeration

File system element type

Syntax

```
enum _PRL_FILE_SYSTEM_ELEMENT_TYPE {
  PSE_DRIVE = 0,
  PSE_DIRECTORY = 1,
  PSE_FILE = 2
};
```

File

PrlEnums.h

_PRL_FILE_SYSTEM_FS_TYPE Enumeration

Syntax

```
enum _PRL_FILE_SYSTEM_FS_TYPE {
  PRL_FS_INVALID = 0,
  PRL_FS_FAT = 1,
  PRL_FS_FAT32 = 2,
  PRL_FS_NTFS = 3,
  PRL_FS_HFS = 4,
  PRL_FS_ADFS = 5,
  PRL_FS_AFFS = 6,
  PRL_FS_AFS = 7,
  PRL_FS_AUTOFS = 8,
  PRL_FS_CODA = 9,
  PRL_FS_EFS = 10,
  PRL_FS_EXTFS = 11,
```

```
    PRL_FS_HPFS = 12,
    PRL_FS_ISOFS = 13,
    PRL_FS_JFFS2 = 14,
    PRL_FS_NFS = 15,
    PRL_FS_QNX4 = 16,
    PRL_FS_REISERFS = 17,
    PRL_FS_SMBFS = 18,
    PRL_FS_GFS = 19,
    PRL_FS_FUSE = 20,
    PRL_FS_UNSPECIFIED = 0xFF
  };
```

File

PrlEnums.h

_PRL_FILE_SYSTEM_TYPE Enumeration

File system general type

Syntax

```
  enum _PRL_FILE_SYSTEM_TYPE {
    PFS_WINDOWS_LIKE_FS = 0,
    PFS_UNIX_LIKE_FS = 1
  };
```

File

PrlEnums.h

_PRL_FIREWALL_DIRECTION Enumeration

Firewall direction

Syntax

```
  enum _PRL_FIREWALL_DIRECTION {
    PFD_INCOMING = 0,
    PFD_OUTGOING = 1
  };
```

File

PrlEnums.h

_PRL_FIREWALL_POLICY Enumeration

Firewall policy

Syntax

```
  enum _PRL_FIREWALL_POLICY {
    PFP_ACCEPT = 0,
    PFP_DENY = 1
  };
```

File

PrlEnums.h

_PRL_GENERIC_DEVICE_STATE Enumeration

Generic device state PGS_CONNECTED_TO_HOST : device is used by host PGS_CONNECTED_TO_VM : device is used by VM PGS_RESERVED : reserved value PGS_NON_CONTROLLED_USB : USB device is not controlled PGS_CONNECTING_TO_VM : device belong to VM, but prl_usb_dev still not started

Syntax

```
enum _PRL_GENERIC_DEVICE_STATE {
  PGS_CONNECTED_TO_HOST = 0,
  PGS_CONNECTED_TO_VM = 1,
  PGS_RESERVED = 2,
  PGS_NON_CONTROLLED_USB = PGS_RESERVED,
  PGS_CONNECTING_TO_VM = 3
};
```

File

PrlEnums.h

_PRL_GENERIC_PCI_DEVICE_CLASS Enumeration

Generic PCI device class: PGD_PCI_NETWORK : generic PCI network device PGD_PCI_DISPLAY : generic PCI display device PGD_PCI_SOUND : generic PCI sound device PGD_PCI_OTHER : another generic PCI device

Syntax

```
enum _PRL_GENERIC_PCI_DEVICE_CLASS {
  PGD_PCI_NETWORK = 0,
  PGD_PCI_DISPLAY = 1,
  PGD_PCI_SOUND = 2,
  PGD_PCI_OTHER = 3
};
```

File

PrlEnums.h

_PRL_GET_SNAPSHOT_TREE_FLAGS Enumeration

PrlVm_GetSnapshotsTreeEx (p. 468) flags

Syntax

```
enum _PRL_GET_SNAPSHOT_TREE_FLAGS {
  PGST_FULL_SIZE_SCREENSHOTS = 1<<(PACF_MAX+1),
  PGST_WITHOUT_SCREENSHOTS = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

Members

PGST_FULL_SIZE_SCREENSHOTS
return screenshots in full size

PGST_WITHOUT_SCREENSHOTS
fast way to request snapshots tree

_PRL_GET_VM_CONFIG_FLAGS Enumeration

Find VM operations:

Syntax

```
enum _PRL_GET_VM_CONFIG_FLAGS {
  PGVC_SEARCH_BY_UUID = 1<<(PACF_MAX+1),
  PGVC_SEARCH_BY_NAME = 1<<(PACF_MAX+2),
  PGVC_FILL_AUTOGENERATED = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_GET_VM_LIST_FLAGS Enumeration

List operations: Uses for PrlSrv_GetVmListEx (p. 266) operation

Syntax

```
enum _PRL_GET_VM_LIST_FLAGS {
  PGVLF_GET_ONLY_VM = PVTF_VM,
  PGVLF_GET_ONLY_CT = PVTF_CT,
  PGVLF_GET_ONLY_IDENTITY_INFO = 1<<(PACF_MAX+3),
  PGVLF_GET_STATE_INFO = 1<<(PACF_MAX+4),
  PGVLF_FILL_AUTOGENERATED = 1<<(PACF_MAX+5),
  PGVLF_GET_ONLY_VM_TEMPLATES = 1<<(PACF_MAX+6)
};
```

File

PrlCommandsFlags.h

_PRL_GRACEFUL_SHUTDOWN_CAPABILITIES Enumeration

Syntax

```
enum _PRL_GRACEFUL_SHUTDOWN_CAPABILITIES {
  PGST_CAPABILITY_SHUTDOWN = 0x1,
  PGST_CAPABILITY_REBOOT = 0x2,
  PGST_CAPABILITY_SUSPEND = 0x4,
```

```
  PGST_CAPABILITY_HIBERNATE = 0x8
};
```

File

PrlEnums.h

_PRL_GRACEFUL_SHUTDOWN_KIND Enumeration

Graceful Shutdown types

Syntax

```
enum _PRL_GRACEFUL_SHUTDOWN_KIND {
  PST_VM_SHUTDOWN = 0,
  PST_VM_REBOOT = 1,
  PST_VM_HIBERNATE = 2,
  PST_VM_SUSPEND = 3
};
```

File

PrlEnums.h

_PRL_GUEST_DISTRO_VERSION_ENUM Enumeration

Windows installation distributive type.

Syntax

```
enum _PRL_GUEST_DISTRO_VERSION_ENUM {
  PGD_UNKNOWN = 0,
  PGD_WINDOWS_VISTA = 1,
  PGD_WINDOWS_XP = 2,
  PGD_WINDOWS_XP_32 = PGD_WINDOWS_XP,
  PGD_WINDOWS_XP_64 = 3,
  PGD_WINDOWS_VISTA_32 = PGD_WINDOWS_VISTA,
  PGD_WINDOWS_VISTA_64 = 4
};
```

File

PrlEnums.h

_PRL_GUEST_OS_SUPPORT_TYPE Enumeration

Specifies the type of guest OS's support

Syntax

```
enum _PRL_GUEST_OS_SUPPORT_TYPE {
  PGS_OS_SUPPORTED = 0,
  PGS_OS_EXPERIMENTAL = 1
};
```

File

PrlEnums.h

_PRL_HANDLE_TYPE Enumeration

The _PRL_HANDLE_TYPE enumeration defines handle types.

Syntax

```
enum _PRL_HANDLE_TYPE {
  PHT_ERROR = 0,
  PHT_SERVER = 0x10000001,
  PHT_SERVER_CONFIG = 0x10000002,
  PHT_REMOTE_FILESYSTEM_INFO = 0x10000003,
  PHT_REMOTE_FILESYSTEM_ENTRY = 0x10000004,
  PHT_VIRTUAL_MACHINE = 0x10000005,
  PHT_VIRTUAL_DEV_DISPLAY = 0x10000006,
  PHT_VIRTUAL_DEV_KEYBOARD = 0x10000007,
  PHT_VIRTUAL_DEV_MOUSE = 0x10000008,
  PHT_VIRTUAL_DEV_FLOPPY = 0x10000009,
  PHT_VIRTUAL_DEV_HARD_DISK = 0x1000000A,
  PHT_VIRTUAL_DEV_NET_ADAPTER = 0x1000000B,
  PHT_VIRTUAL_DEV_PARALLEL_PORT = 0x1000000C,
  PHT_VIRTUAL_DEV_SERIAL_PORT = 0x1000000D,
  PHT_VIRTUAL_DEV_OPTICAL_DISK = 0x1000000E,
  PHT_VIRTUAL_DEV_USB_DEVICE = 0x1000000F,
  PHT_VIRTUAL_DEV_SOUND = 0x10000010,
  PHT_JOB = 0x10000011,
  PHT_EVENT = 0x10000012,
  PHT_RESULT = 0x10000013,
  PHT_VM_INFO = 0x10000014,
  PHT_USER_PROFILE = 0x10000015,
  PHT_DISP_CONFIG = 0x10000016,
  PHT_SHARE = 0x10000017,
  PHT_SCREEN_RESOLUTION = 0x10000018,
  PHT_DISP_NET_ADAPTER = 0x10000019,
  PHT_HW_GENERIC_DEVICE = 0x1000001A,
  PHT_HW_HARD_DISK = 0x1000001B,
  PHT_HW_HARD_DISK_PARTITION = 0x1000001C,
  PHT_HW_NET_ADAPTER = 0x1000001D,
  PHT_EVENT_PARAMETER = 0x1000001E,
  PHT_REMOTEDEV_CMD = 0x1000001F,
  PHT_SYSTEM_STATISTICS = 0x10000020,
  PHT_SYSTEM_STATISTICS_CPU = 0x10000021,
  PHT_SYSTEM_STATISTICS_IFACE = 0x10000022,
  PHT_SYSTEM_STATISTICS_USER_SESSION = 0x10000023,
  PHT_SYSTEM_STATISTICS_DISK = 0x10000024,
  PHT_SYSTEM_STATISTICS_DISK_PARTITION = 0x10000025,
  PHT_SYSTEM_STATISTICS_PROCESS = 0x10000026,
  PHT_LICENSE = 0x10000027,
  PHT_SERVER_INFO = 0x10000028,
  PHT_BOOT_DEVICE = 0x10000029,
  PHT_STRINGS_LIST = 0x1000002A,
  PHT_LOGIN_RESPONSE = 0x1000002B,
  PHT_RUNNING_TASK = 0x1000002C,
  PHT_NET_SERVICE_STATUS = 0x1000002D,
  PHT_HANDLES_LIST = 0x1000002E,
  PHT_VIRTUAL_DISK = 0x1000002F,
  PHT_ACCESS_RIGHTS = 0x10000030,
  PHT_VM_TOOLS_INFO = 0x10000031,
  PHT_FOUND_VM_INFO = 0x10000032,
```

```
    PHT_TIS_RECORD = 0x10000033,
    PHT_USER_INFO = 0x10000034,
    PHT_VM_CONFIGURATION = PHT_VIRTUAL_MACHINE,
    PHT_TIS_EMITTER = 0x10000036,
    PHT_TOOL = 0x10000037,
    PHT_VIRTUAL_DEV_HD_PARTITION = 0x10000038,
    PHT_VIRTUAL_DEV_GENERIC_PCI = 0x10000039,
    PHT_VIRTUAL_DEV_GENERIC_SCSI = 0x1000003A,
    PHT_VM_GUEST_SESSION = 0x1000003B,
    PHT_VIRTUAL_NETWORK = 0x1000003C,
    PHT_PORT_FORWARDING = 0x1000003D,
    PHT_HW_GENERIC_PCI_DEVICE = 0x1000003E,
    PHT_OPAQUE_TYPE_LIST = 0x1000003F,
    PHT_GUEST_OSES_MATRIX = 0x10000040,
    PHT_PROBLEM_REPORT = 0x10000041,
    PHT_APPLIANCE_CONFIG = 0x10000042,
    PHT_OFFLINE_SERVICE = 0x10000043,
    PHT_VM_VIRTUAL_DEVICES_INFO = 0x10000044,
    PHT_NETWORK_CLASS = 0x10000045,
    PHT_NETWORK_SHAPING = 0x10000046,
    PHT_NETWORK_RATE = 0x10000047,
    PHT_ISCSI_LUN = 0x10000048,
    PHT_CT_TEMPLATE = 0x10000049,
    PHT_USB_IDENTITY = 0x10000051,
    PHT_FIREWALL_RULE = 0x10000052,
    PHT_IPPRIV_NET = 0x10000053,
    PHT_PLUGIN_INFO = 0x10000054,
    PHT_CVSRC = 0x10000055,
    PHT_DESKTOP_CONTROL = 0x10000056,
    PHT_BACKUP_RESULT = 0x10000057,
    PHT_NETWORK_SHAPING_CONFIG = 0x10000058,
    PHT_NETWORK_SHAPING_BANDWIDTH = 0x10000059,
    PHT_SYSTEM_STATISTICS_VM_DATA = 0x1000005A,
    PHT_VIDEO_RECEIVER = 0x1000005B,
    PHT_SESSION_INFO = 0x1000005C,
    PHT_CPU_FEATURES = 0x1000005D,
    PHT_CPU_POOL = 0x1000005E,
    PHT_VIRTUAL_DISK_MAP = 0x1000005F,
    PHT_VIRTUAL_DISK_OPEN_POLICY = 0x10000060,
    PHT_VIRTUAL_DISK_ENCRYPTION = 0x10000061,
    PHT_VCMMD_CONFIG = 0x10000062,
    PHT_LAST = PHT_VCMMD_CONFIG
};
```

File

PrlEnums.h

Members

PHT_LAST
should be set to last

Remarks

Each handle type and the functions that operate on it is described in detail in the "Functions by Handle" chapter of the Virtuozzo C API Reference guide.

_PRL_HARD_DISK_IFACE_MASK Enumeration

Hard disk interfce mask enum

Syntax

```
enum _PRL_HARD_DISK_IFACE_MASK {
  PIM_IDE_0_0 = (0x1L<<0),
  PIM_IDE_0_1 = (0x1L<<1),
  PIM_IDE_1_0 = (0x1L<<2),
  PIM_IDE_1_1 = (0x1L<<3),
  PIM_SCSI_0_0 = (0x1L<<4),
  PIM_SCSI_1_0 = (0x1L<<5),
  PIM_SCSI_2_0 = (0x1L<<6),
  PIM_SCSI_3_0 = (0x1L<<7),
  PIM_SCSI_4_0 = (0x1L<<8),
  PIM_SCSI_5_0 = (0x1L<<9),
  PIM_SCSI_6_0 = (0x1L<<10),
  PIM_SCSI_7_0 = (0x1L<<11),
  PIM_SCSI_8_0 = (0x1L<<12),
  PIM_SCSI_9_0 = (0x1L<<13),
  PIM_SCSI_10_0 = (0x1L<<14),
  PIM_SCSI_11_0 = (0x1L<<15),
  PIM_SCSI_12_0 = (0x1L<<16),
  PIM_SCSI_13_0 = (0x1L<<17),
  PIM_SCSI_14_0 = (0x1L<<18),
  PIM_SCSI_15_0 = (0x1L<<19),
  PIM_SATA_0_0 = (0x1L<<20),
  PIM_SATA_0_1 = (0x1L<<21),
  PIM_SATA_0_2 = (0x1L<<22),
  PIM_SATA_0_3 = (0x1L<<23),
  PIM_SATA_0_4 = (0x1L<<24),
  PIM_SATA_0_5 = (0x1L<<25),
  PIM_ALL_DISKS = ((0x1L<<26) - 1),
  PIM_COMPACT_DELAYED = (0x1L<<31),
  PIM_IDE_MASK_OFFSET = PIM_IDE_0_0,
  PIM_SCSI_MASK_OFFSET = PIM_SCSI_0_0,
  PIM_SATA_MASK_OFFSET = PIM_SATA_0_0
};
```

File

PrlEnums.h

_PRL_HARD_DISK_INTERNAL_FORMAT Enumeration

Hard disk image type

Syntax

```
enum _PRL_HARD_DISK_INTERNAL_FORMAT {
  PHD_PLAIN_HARD_DISK = 0,
  PHD_EXPANDING_HARD_DISK = 1
};
```

File

PrlEnums.h

_PRL_HOST_OS_TYPE Enumeration

Host OS type

Syntax

```
enum _PRL_HOST_OS_TYPE {
  PHO_MAC = 0,
  PHO_LIN = 1,
  PHO_WIN = 2,
  PHO_UNKNOWN = 0xFF
};
```

File

PrlEnums.h

_PRL_HW_INFO_NET_ADAPTER_TYPE Enumeration

Network adapter types in hardware info list

Syntax

```
enum _PRL_HW_INFO_NET_ADAPTER_TYPE {
  PHI_REAL_NET_ADAPTER = 0,
  PHI_VIRTUAL_NET_ADAPTER = 1,
  PHY_WIFI_REAL_NET_ADAPTER = 2
};
```

File

PrlEnums.h

_PRL_IMAGE_FORMAT Enumeration

Syntax

```
enum _PRL_IMAGE_FORMAT {
  PIF_RAW = 0x50000008,
  PIF_BMP = 0x50000009,
  PIF_JPG = 0x5000000A,
  PIF_PNG = 0x5000000B
};
```

File

PrlEnums.h

_PRL_INSTALL_APPLIANCE_FLAGS Enumeration

Install appliance options

- PIAF_CANCEL : cancel appliance installation
- PIAF_FORCE : do not cancel installation and begin a new one
- if the previous installation became corrupted outside.
- PIAF_STOP : temporary stop appliance downloading

Syntax

```
enum _PRL_INSTALL_APPLIANCE_FLAGS {
  PIAF_CANCEL = 1<<(PACF_MAX+1),
  PIAF_FORCE = 1<<(PACF_MAX+2),
  PIAF_STOP = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_IOLIMIT_TYPE Enumeration

Syntax

```
enum _PRL_IOLIMIT_TYPE {
  PRL_IOLIMIT_BS = 1
};
```

File

PrlEnums.h

_PRL_IOPRIO Enumeration

Syntax

```
enum _PRL_IOPRIO {
  PRL_IOPRIO_MIN = 0,
  PRL_IOPRIO_DEFAULT = 4,
  PRL_IOPRIO_MAX = 7
};
```

File

PrlEnums.h

_PRL_ISCSI_STORAGE_CMD_FLAGS Enumeration

iSCSI storage management flags

Syntax

```
enum _PRL_ISCSI_STORAGE_CMD_FLAGS {
  PISCSI_STORAGE_CREATE = 1<<(PACF_MAX+1),
  PISCSI_STORAGE_REMOVE = 1<<(PACF_MAX+2),
  PISCSI_STORAGE_MOUNT_RDONLY = 1<<(PACF_MAX+3),
  PISCSI_STORAGE_REMOUNT = 1<<(PACF_MAX+4),
  PISCSI_STORAGE_MOUNT = 1<<(PACF_MAX+5)
};
```

File

PrlCommandsFlags.h

Members

PISCSI_STORAGE_CREATE
To create storage on register

PISCSI_STORAGE_REMOVE
To remove storage on unregister

PISCSI_STORAGE_MOUNT_RDONLY
To mount storage by read-only

PISCSI_STORAGE_REMOUNT
To remount storage

PISCSI_STORAGE_MOUNT
To mount storage on register

_PRL_ISCSI_STORAGE_FSTYPE_FLAGS Enumeration

iSCSI storage filesystem type

Syntax

```
enum _PRL_ISCSI_STORAGE_FSTYPE_FLAGS {
  PISCSI_STORAGE_EXT3 = 1<<(PACF_MAX+1),
  PISCSI_STORAGE_EXT4 = 1<<(PACF_MAX+2),
  PISCSI_STORAGE_NTFS = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_JOB_OPERATION_CODE Enumeration

Job operation codes

Syntax

```
enum _PRL_JOB_OPERATION_CODE {
  PJOC_UNKNOWN = 0,
  PJOC_JOB_CANCEL = 1,
  PJOC_SRV_LOOKUP_PARALLELS_SERVERS = 2,
  PJOC_SRV_LOGIN = 3,
  PJOC_SRV_LOGIN_LOCAL = 4,
  PJOC_SRV_LOGOFF = 5,
  PJOC_SRV_GET_SRV_CONFIG = 6,
  PJOC_SRV_GET_COMMON_PREFS = 7,
  PJOC_SRV_COMMON_PREFS_BEGIN_EDIT = 8,
  PJOC_SRV_COMMON_PREFS_COMMIT = 9,
  PJOC_SRV_GET_USER_PROFILE = 10,
  PJOC_SRV_GET_STATISTICS = 11,
  PJOC_SRV_USER_PROFILE_BEGIN_EDIT = 12,
  PJOC_SRV_USER_PROFILE_COMMIT = 13,
  PJOC_SRV_REGISTER_VM = 14,
  PJOC_SRV_REGISTER_VM_EX = PJOC_SRV_REGISTER_VM,
  PJOC_SRV_GET_VM_LIST = 15,
  PJOC_SRV_SUBSCRIBE_TO_HOST_STATISTICS = 16,
```

```
    PJOC_SRV_UNSUBSCRIBE_FROM_HOST_STATISTICS = 17,
    PJOC_SRV_SHUTDOWN = 18,
    PJOC_SRV_FS_GET_DISK_LIST = 19,
    PJOC_SRV_FS_GET_DIR_ENTRIES = 20,
    PJOC_SRV_FS_CREATE_DIR = 21,
    PJOC_SRV_FS_REMOVE_ENTRY = 22,
    PJOC_SRV_FS_CAN_CREATE_FILE = 23,
    PJOC_SRV_FS_RENAME_ENTRY = 24,
    PJOC_SRV_FS_GENERATE_ENTRY_NAME = 25,
    PJOC_SRV_UPDATE_LICENSE = 26,
    PJOC_SRV_GET_LICENSE_INFO = 27,
    PJOC_SRV_SEND_ANSWER = 28,
    PJOC_SRV_START_SEARCH_VMS = 29,
    PJOC_SRV_NET_SERVICE_START = 30,
    PJOC_SRV_NET_SERVICE_STOP = 31,
    PJOC_SRV_NET_SERVICE_RESTART = 32,
    PJOC_SRV_NET_SERVICE_RESTORE_DEFAULTS = 33,
    PJOC_SRV_GET_NET_SERVICE_STATUS = 34,
    PJOC_SRV_ADD_NET_ADAPTER = 35,
    PJOC_SRV_DELETE_NET_ADAPTER = 36,
    PJOC_SRV_UPDATE_NET_ADAPTER = 37,
    PJOC_SRV_GET_PROBLEM_REPORT = 38,
    PJOC_SRV_ATTACH_TO_LOST_TASK = 39,
    PJOC_SRV_GET_USER_INFO_LIST = 40,
    PJOC_SRV_GET_USER_INFO = 41,
    PJOC_VM_START = 42,
    PJOC_VM_STOP = 43,
    PJOC_VM_PAUSE = 44,
    PJOC_VM_RESET = 45,
    PJOC_VM_SUSPEND = 46,
    PJOC_VM_RESUME = PJOC_VM_START,
    PJOC_VM_DROP_SUSPENDED_STATE = 49,
    PJOC_VM_CLONE = 50,
    PJOC_VM_DELETE = 51,
    PJOC_VM_GET_PROBLEM_REPORT = 52,
    PJOC_VM_GET_STATE = 53,
    PJOC_VM_GENERATE_VM_DEV_FILENAME = 54,
    PJOC_VM_GET_TOOLS_STATE = 55,
    PJOC_VM_REFRESH_CONFIG = 56,
    PJOC_VM_GET_STATISTICS = 57,
    PJOC_VM_SUBSCRIBE_TO_GUEST_STATISTICS = 58,
    PJOC_VM_UNSUBSCRIBE_FROM_GUEST_STATISTICS = 59,
    PJOC_VM_REG = 60,
    PJOC_VM_UNREG = 61,
    PJOC_VM_BEGIN_EDIT = 62,
    PJOC_VM_COMMIT = 63,
    PJOC_VM_CREATE_UNATTENDED_FLOPPY = 64,
    PJOC_VM_INITIATE_DEV_STATE_NOTIFICATIONS = 65,
    PJOC_VM_UPDATE_SECURITY = 66,
    PJOC_VM_VALIDATE_CONFIG = 67,
    PJOC_VM_DEV_CONNECT = 68,
    PJOC_VM_DEV_DISCONNECT = 69,
    PJOC_VM_DEV_CREATE_IMAGE = 70,
    PJOC_VM_INSTALL_TOOLS = 71,
    PJOC_VM_SEND_SHUTDOWN_COMMAND = 72,
    PJOC_VM_SEND_CLIPBOARD_REQUEST = 73,
    PJOC_VM_CONNECT_TO_VM = 74,
    PJOC_VM_DEV_DISPLAY_CAPTURE_SCREEN = 75,
    PJOC_VM_MIGRATE = 76,
    PJOC_VM_SEND_DRAGDROP_COMMAND = 77,
    PJOC_VM_GUEST_RUN_PROGRAM = 78,
    PJOC_VM_GUEST_LOGOUT = 79,
    PJOC_SRV_SUBSCRIBE_PERFSTATS = 80,
    PJOC_SRV_UNSUBSCRIBE_PERFSTATS = 81,
    PJOC_SRV_GET_PERFSTATS = 82,
    PJOC_VM_SUBSCRIBE_PERFSTATS = 83,
```

```
    PJOC_VM_UNSUBSCRIBE_PERFSTATS = 84,
    PJOC_VM_GET_PERFSTATS = 85,
    PJOC_VM_GET_SUSPENDED_SCREEN = 86,
    PJOC_VM_MIGRATE_CANCEL = 87,
    PJOC_VM_RUN_COMPRESSOR = 88,
    PJOC_VM_CANCEL_COMPRESSOR = 89,
    PJOC_VM_CREATE_SNAPSHOT = 90,
    PJOC_VM_SWITCH_TO_SNAPSHOT = 91,
    PJOC_VM_DELETE_SNAPSHOT = 92,
    PJOC_VM_GET_SNAPSHOTS_TREE = 93,
    PJOC_VM_START_EX = 94,
    PJOC_VM_LOGIN_IN_GUEST = 95,
    PJOC_VM_UPDATE_SNAPSHOT_DATA = 96,
    PJOC_VM_RESTART = 97,
    PJOC_VM_INSTALL_UTILITY = 98,
    PJOC_VM_UPDATE_TOOLS_SECTION = 99,
    PJOC_SRV_GET_ALL_HOST_USERS = 100,
    PJOC_VM_RESTORE = 101,
    PJOC_VM_START_VNC_SERVER = 102,
    PJOC_VM_STOP_VNC_SERVER = 103,
    PJOC_VM_GUEST_GET_NETWORK_SETTINGS = 104,
    PJOC_SRV_GET_VIRTUAL_NETWORK_LIST = 105,
    PJOC_VM_AUTH_WITH_GUEST_SECURITY_DB = 106,
    PJOC_VM_GUEST_SET_USER_PASSWD = 107,
    PJOC_SRV_ADD_VIRTUAL_NETWORK = 108,
    PJOC_SRV_UPDATE_VIRTUAL_NETWORK = 109,
    PJOC_SRV_DELETE_VIRTUAL_NETWORK = 110,
    PJOC_SRV_CONFIGURE_GENERIC_PCI = 111,
    PJOC_SRV_PREPARE_FOR_HIBERNATE = 112,
    PJOC_SRV_AFTER_HOST_RESUME = 113,
    PJOC_SRV_GET_BACKUP_TREE = 114,
    PJOC_SRV_CREATE_VM_BACKUP = 115,
    PJOC_SRV_RESTORE_VM_BACKUP = 116,
    PJOC_API_SEND_PROBLEM_REPORT = 117,
    PJOC_VM_LOCK = 118,
    PJOC_VM_UNLOCK = 119,
    PJOC_SRV_SET_NON_INTERACTIVE_SESSION = 120,
    PJOC_SRV_CREATE_UNATTENDED_CD = 121,
    PJOC_SRV_REMOVE_VM_BACKUP = 122,
    PJOC_VM_RESIZE_DISK_IMAGE = 123,
    PJOC_SRV_SET_SESSION_CONFIRMATION_MODE = 124,
    PJOC_SRV_STORE_VALUE_BY_KEY = 125,
    PJOC_VM_STORE_VALUE_BY_KEY = 126,
    PJOC_VM_COMPACT = 127,
    PJOC_VM_CANCEL_COMPACT = 128,
    PJOC_SRV_SEND_CLIENT_STATISTICS = 129,
    PJOC_VM_GET_PACKED_PROBLEM_REPORT = 130,
    PJOC_SRV_GET_PACKED_PROBLEM_REPORT = 131,
    PJOC_API_SEND_PACKED_PROBLEM_REPORT = 132,
    PJOC_SRV_UPDATE_USB_ASSOC_LIST = 133,
    PJOC_SRV_REGISTER_3RD_PARTY_VM = 134,
    PJOC_VM_CHANGE_SID = 135,
    PJOC_VM_RESET_UPTIME = 136,
    PJOC_REPORT_ASSEMBLY = 137,
    PJOC_SRV_INSTALL_APPLIANCE = 138,
    PJOC_SRV_UPDATE_OFFLINE_SERVICE = 139,
    PJOC_SRV_DELETE_OFFLINE_SERVICE = 140,
    PJOC_SRV_GET_OFFLINE_SERVICES_LIST = 141,
    PJOC_SRV_START_CLUSTER_SERVICE = 142,
    PJOC_SRV_STOP_CLUSTER_SERVICE = 143,
    PJOC_VM_CMD_INTERNAL = 144,
    PJOC_VM_GET_VIRTUAL_DEVICES_INFO = 145,
    PJOC_VM_CONVERT_DISKS = 146,
    PJOC_SRV_UPDATE_NETWORK_CLASSES_CONFIG = 152,
    PJOC_SRV_GET_NETWORK_CLASSES_LIST = 153,
    PJOC_SRV_UPDATE_NETWORK_SHAPING_CONFIG = 154,
```

```
    PJOC_SRV_GET_NETWORK_SHAPING_CONFIG = 155,
    PJOC_SRV_RESTART_NETWORK_SHAPING = 156,
    PJOC_SRV_PROXY_GET_REGISTERED_HOSTS = 157,
    PJOC_SRV_REGISTER_ISCSI_STORAGE = 158,
    PJOC_SRV_UNREGISTER_ISCSI_STORAGE = 159,
    PJOC_SRV_EXTEND_ISCSI_STORAGE = 160,
    PJOC_SRV_GET_CT_TEMPLATE_LIST = 161,
    PJOC_SRV_CHECK_ALIVE = 162,
    PJOC_SRV_GET_DEFAULT_VM_CONFIG = 163,
    PJOC_SRV_REMOVE_CT_TEMPLATE = 164,
    PJOC_SRV_COPY_CT_TEMPLATE = 165,
    PJOC_SRV_ADD_IPPRIVATE_NETWORK = 166,
    PJOC_SRV_REMOVE_IPPRIVATE_NETWORK = 167,
    PJOC_SRV_UPDATE_IPPRIVATE_NETWORK = 168,
    PJOC_SRV_GET_IPPRIVATE_NETWORKS_LIST = 169,
    PJOC_VM_DEV_COPY_IMAGE = 170,
    PJOC_SRV_REFRESH_PLUGINS = 171,
    PJOC_VM_MOUNT = 172,
    PJOC_VM_UMOUNT = 173,
    PJOC_SRV_GET_PLUGINS_LIST = 174,
    PJOC_SRV_GET_DISK_FREE_SPACE = 175,
    PJOC_DESKTOP_CONTROL_CONNECT = 176,
    PJOC_VM_MOVE = 177,
    PJOC_SRV_GENERATE_CSR = 178,
    PJOC_SRV_SIGN_CERTIFICATION_REQUEST = 179,
    PJOC_SRV_GET_VM_CONFIG = 180,
    PJOC_PTM_RPC = 181,
    PJOC_API_PUT_USER_DATA = 182,
    PJOC_API_GET_USER_DATA = 183,
    PJOC_API_GET_USER_KEYS = 184,
    PJOC_API_REMOVE_USER_DATA = 185,
    PJOC_API_PUT_ICON = 186,
    PJOC_API_VALIDATE_IN_CACHE = 187,
    PJOC_SRV_UPDATE_SESSION_INFO = 188,
    PJOC_API_SUBSCRIBE_ON_REGISTRATION = 189,
    PJOC_SRV_SEND_PROBLEM_REPORT = 190,
    PJOC_VM_SEND_PROBLEM_REPORT = 191,
    PJOC_VM_SET_PROTECTION = 192,
    PJOC_VM_REMOVE_PROTECTION = 193,
    PJOC_SRV_CPU_POOLS_LIST_POOLS = 194,
    PJOC_SRV_CPU_POOLS_MOVE = 195,
    PJOC_SRV_CPU_POOLS_RECALCULATE = 196,
    PJOC_SRV_CPU_POOLS_JOIN = 197,
    PJOC_SRV_CPU_POOLS_LEAVE = 198,
    PJOC_VM_COMMIT_ENCRYPTION = 199,
    PJOC_SRV_GET_VCMMD_CONFIG = 200,
    PJOC_CT_REINSTALL = 201
  };
```

File

PrlEnums.h

_PRL_JOB_STATUS Enumeration

Job status enumeration

Syntax

```
  enum _PRL_JOB_STATUS {
    PJS_UNKNOWN = 0,
    PJS_RUNNING = 0xD0000001,
    PJS_FINISHED = 0xD0000002
```

```
};
```

File

PrlEnums.h

_PRL_KEY Enumeration

Virtuozzo virtual keycodes.

Syntax

```
enum _PRL_KEY {
  PRL_KEY_INVALID = 0,
  PRL_KEY_WILDCARD_ANY = 1,
  PRL_KEY_WILDCARD_KEYBOARD = 2,
  PRL_KEY_WILDCARD_MOUSE = 3,
  PRL_KEY_WILDCARD_SHIFT = 4,
  PRL_KEY_WILDCARD_CTRL = 5,
  PRL_KEY_WILDCARD_ALT = 6,
  PRL_KEY_WILDCARD_WIN = 7,
  PRL_KEY_ESCAPE = 9,
  PRL_KEY_1 = 10,
  PRL_KEY_2 = 11,
  PRL_KEY_3 = 12,
  PRL_KEY_4 = 13,
  PRL_KEY_5 = 14,
  PRL_KEY_6 = 15,
  PRL_KEY_7 = 16,
  PRL_KEY_8 = 17,
  PRL_KEY_9 = 18,
  PRL_KEY_0 = 19,
  PRL_KEY_MINUS = 20,
  PRL_KEY_EQUAL = 21,
  PRL_KEY_BACKSPACE = 22,
  PRL_KEY_TAB = 23,
  PRL_KEY_Q = 24,
  PRL_KEY_W = 25,
  PRL_KEY_E = 26,
  PRL_KEY_R = 27,
  PRL_KEY_T = 28,
  PRL_KEY_Y = 29,
  PRL_KEY_U = 30,
  PRL_KEY_I = 31,
  PRL_KEY_O = 32,
  PRL_KEY_P = 33,
  PRL_KEY_LEFT_BRACKET = 34,
  PRL_KEY_RIGHT_BRACKET = 35,
  PRL_KEY_ENTER = 36,
  PRL_KEY_LEFT_CONTROL = 37,
  PRL_KEY_A = 38,
  PRL_KEY_S = 39,
  PRL_KEY_D = 40,
  PRL_KEY_F = 41,
  PRL_KEY_G = 42,
  PRL_KEY_H = 43,
  PRL_KEY_J = 44,
  PRL_KEY_K = 45,
  PRL_KEY_L = 46,
  PRL_KEY_SEMICOLON = 47,
  PRL_KEY_QUOTE = 48,
  PRL_KEY_TILDA = 49,
  PRL_KEY_LEFT_SHIFT = 50,
```

```
    PRL_KEY_BACKSLASH = 51,
    PRL_KEY_Z = 52,
    PRL_KEY_X = 53,
    PRL_KEY_C = 54,
    PRL_KEY_V = 55,
    PRL_KEY_B = 56,
    PRL_KEY_N = 57,
    PRL_KEY_M = 58,
    PRL_KEY_COMMA = 59,
    PRL_KEY_DOT = 60,
    PRL_KEY_SLASH = 61,
    PRL_KEY_RIGHT_SHIFT = 62,
    PRL_KEY_NP_STAR = 63,
    PRL_KEY_LEFT_ALT = 64,
    PRL_KEY_SPACE = 65,
    PRL_KEY_CAPS_LOCK = 66,
    PRL_KEY_F1 = 67,
    PRL_KEY_F2 = 68,
    PRL_KEY_F3 = 69,
    PRL_KEY_F4 = 70,
    PRL_KEY_F5 = 71,
    PRL_KEY_F6 = 72,
    PRL_KEY_F7 = 73,
    PRL_KEY_F8 = 74,
    PRL_KEY_F9 = 75,
    PRL_KEY_F10 = 76,
    PRL_KEY_NUM_LOCK = 77,
    PRL_KEY_SCROLL_LOCK = 78,
    PRL_KEY_NP_7 = 79,
    PRL_KEY_NP_8 = 80,
    PRL_KEY_NP_9 = 81,
    PRL_KEY_NP_MINUS = 82,
    PRL_KEY_NP_4 = 83,
    PRL_KEY_NP_5 = 84,
    PRL_KEY_NP_6 = 85,
    PRL_KEY_NP_PLUS = 86,
    PRL_KEY_NP_1 = 87,
    PRL_KEY_NP_2 = 88,
    PRL_KEY_NP_3 = 89,
    PRL_KEY_NP_0 = 90,
    PRL_KEY_NP_DELETE = 91,
    PRL_KEY_PRINT = 92,
    PRL_KEY_EUROPE_1 = 93,
    PRL_KEY_EUROPE_2 = 94,
    PRL_KEY_F11 = 95,
    PRL_KEY_F12 = 96,
    PRL_KEY_HOME = 97,
    PRL_KEY_UP = 98,
    PRL_KEY_PAGE_UP = 99,
    PRL_KEY_LEFT = 100,
    PRL_KEY_RIGHT = 102,
    PRL_KEY_END = 103,
    PRL_KEY_DOWN = 104,
    PRL_KEY_PAGE_DOWN = 105,
    PRL_KEY_INSERT = 106,
    PRL_KEY_DELETE = 107,
    PRL_KEY_NP_ENTER = 108,
    PRL_KEY_RIGHT_CONTROL = 109,
    PRL_KEY_PAUSE = 110,
    PRL_KEY_NP_SLASH = 112,
    PRL_KEY_RIGHT_ALT = 113,
    PRL_KEY_LEFT_WIN = 115,
    PRL_KEY_RIGHT_WIN = 116,
    PRL_KEY_MENU = 117,
    PRL_KEY_MEDIA_NEXT_TRACK = 118,
    PRL_KEY_MEDIA_PREV_TRACK = 119,
```

```
    PRL_KEY_MEDIA_STOP = 120,
    PRL_KEY_MEDIA_PLAY_PAUSE = 121,
    PRL_KEY_MUTE = 122,
    PRL_KEY_VOLUME_UP = 123,
    PRL_KEY_VOLUME_DOWN = 124,
    PRL_KEY_MEDIA_SELECT = 125,
    PRL_KEY_APP_MAIL = 126,
    PRL_KEY_APP_CALCULATOR = 127,
    PRL_KEY_APP_MY_COMPUTER = 128,
    PRL_KEY_WWW_SEARCH = 129,
    PRL_KEY_WWW_HOME = 130,
    PRL_KEY_WWW_BACK = 131,
    PRL_KEY_WWW_FORWARD = 132,
    PRL_KEY_WWW_STOP = 133,
    PRL_KEY_WWW_REFRESH = 134,
    PRL_KEY_WWW_FAVORITES = 135,
    PRL_KEY_EJECT = 136,
    PRL_KEY_SYSTEM_POWER = 137,
    PRL_KEY_SYSTEM_SLEEP = 138,
    PRL_KEY_SYSTEM_WAKE = 139,
    PRL_KEY_BRAZILIAN_KEYPAD = 140,
    PRL_KEY_RO = 141,
    PRL_KEY_HIRAGANA_KATAKANA = 142,
    PRL_KEY_YEN = 143,
    PRL_KEY_HENKAN = 144,
    PRL_KEY_MUHENKAN = 145,
    PRL_KEY_PC9800_KEYPAD = 146,
    PRL_KEY_HANGUEL = 147,
    PRL_KEY_HANJA = 148,
    PRL_KEY_KATAKANA = 149,
    PRL_KEY_HIRAGANA = 150,
    PRL_KEY_ZENKAKU_HANKAKU = 151,
    PRL_KEY_F13 = 152,
    PRL_KEY_F14 = 153,
    PRL_KEY_F15 = 154,
    PRL_KEY_F16 = 155,
    PRL_KEY_F17 = 156,
    PRL_KEY_F18 = 157,
    PRL_KEY_F19 = 158,
    PRL_KEY_F20 = 159,
    PRL_KEY_F21 = 160,
    PRL_KEY_F22 = 161,
    PRL_KEY_F23 = 162,
    PRL_KEY_F24 = 163,
    PRL_KEY_NP_EQUAL = 164,
    PRL_KEY_BREAK = 165,
    PRL_KEY_PRINT_WITH_MODIFIER = 166,
    PRL_KEY_SYSRQ = 167,
    PRL_KEY_FN = 168,
    PRL_KEY_EURO = 169,
    PRL_KEY_DOLLAR = 170,
    PRL_KEY_MON_BRIGHTNESS_DOWN = 171,
    PRL_KEY_MON_BRIGHTNESS_UP = 172,
    PRL_KEY_APP_EXPOSE = 173,
    PRL_KEY_APP_DASHBOARD = 174,
    PRL_KEY_KBD_BRIGHTNESS_DOWN = 175,
    PRL_KEY_KBD_BRIGHTNESS_UP = 176,
    PRL_KEY_MAC129 = 177,
    PRL_KEY_LEFT_BUTTON = 178,
    PRL_KEY_MIDDLE_BUTTON = 179,
    PRL_KEY_RIGHT_BUTTON = 180,
    PRL_KEY_MOVE_UP_LEFT = 181,
    PRL_KEY_MOVE_UP = 182,
    PRL_KEY_MOVE_UP_RIGHT = 183,
    PRL_KEY_MOVE_LEFT = 184,
    PRL_KEY_MOVE_RIGHT = 185,
```

```
    PRL_KEY_MOVE_DOWN_LEFT = 186,
    PRL_KEY_MOVE_DOWN = 187,
    PRL_KEY_MOVE_DOWN_RIGHT = 188,
    PRL_KEY_WHEEL_UP = 189,
    PRL_KEY_WHEEL_DOWN = 190,
    PRL_KEY_WHEEL_LEFT = 191,
    PRL_KEY_WHEEL_RIGHT = 192,
    PRL_KEY_MAX = 193
  };
```

File

PrlKeys.h

_PRL_KEY_EVENT Enumeration

Syntax

```
  enum _PRL_KEY_EVENT {
    PKE_PRESS = 0x0,
    PKE_RELEASE = 0x80
  };
```

File

PrlEnums.h

_PRL_LICENSE_RESTRICTION_KEY Enumeration

License restriction options

Syntax

```
  enum _PRL_LICENSE_RESTRICTION_KEY {
    PLRK_UNKNOWN = 0,
    PLRK_VM_CPU_LIMIT = 1,
    PLRK_VM_MEMORY_LIMIT = 2,
    PLRK_VM_VTD_AVAILABLE = 3,
    PLRK_ALLOWED_GUEST_OS_VERSIONS = 4,
    PLRK_RUNNING_VMS_LIMIT = 5,
    PLRK_VM_CREATE = 6,
    PLRK_VM_REGISTER = 7,
    PLRK_VM_IMPORT_3RD_PARTY = 8,
    PLRK_VM_CLONE = 9,
    PLRK_VM_CONVERT_TO_TEMPLATE = 10,
    PLRK_VM_CONVERT_FROM_TEMPLATE = 11,
    PLRK_VM_DOWNLOAD = 12,
    PLRK_VM_SUSPEND = 13,
    PLRK_VM_PAUSE = 14,
    PLRK_VM_SNAPSHOT_CREATE = 15,
    PLRK_VM_SNAPSHOT_SWITCH = 16,
    PLRK_VM_SNAPSHOT_DELETE = 17,
    PLRK_VM_SNAPSHOT_TREE = 18,
    PLRK_VM_UNDODISK_FEATURE = 19,
    PLRK_VM_SAFEMODE_FEATURE = 20,
    PLRK_VM_SMARTGUARD_FEATURE = 21,
    PLRK_VM_SHOW_FULLSCREEN = 22,
    PLRK_RUNNING_VMS_LIMIT_PER_USER = 23
  };
```

File

PrlEnums.h

Members

PLRK_UNKNOWN
NOTE to Virtuozzo Developers:

- DO NOT CHANGE ANY MEMBER NAME BECAUSE IT USES IN INTERNAL PROTOCOL(ParallelsDomModel.h)

PLRK_VM_CPU_LIMIT
Value is PRL_UINT32 (p. 798)

PLRK_VM_MEMORY_LIMIT
Value is PRL_UINT32 (p. 798) in Mb

PLRK_VM_VTD_AVAILABLE
Value is PRL_UINT32 (p. 798)

PLRK_ALLOWED_GUEST_OS_VERSIONS
Value is PHT_OPAQUE_TYPE_LIST (p. 177) of Os version values from PrlOses.h

PLRK_RUNNING_VMS_LIMIT
Value is PRL_UINT32 (p. 798) (max count on the host)

PLRK_VM_CREATE
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_REGISTER
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_IMPORT_3RD_PARTY
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_CLONE
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_CONVERT_TO_TEMPLATE
Value is PRL_UINT32 (p. 798)

PLRK_VM_CONVERT_FROM_TEMPLATE
Value is PRL_UINT32 (p. 798)

PLRK_VM_DOWNLOAD
Value is PRL_UINT32 (p. 798)

PLRK_VM_SUSPEND
Value is PRL_UINT32 (p. 798)

PLRK_VM_PAUSE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_CREATE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_SWITCH
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_DELETE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_TREE
Value is PRL_UINT32 (p. 798)

PLRK_VM_UNDODISK_FEATURE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SAFEMODE_FEATURE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SMARTGUARD_FEATURE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SHOW_FULLSCREEN
Value is PRL_UINT32 (p. 798)

PLRK_RUNNING_VMS_LIMIT_PER_USER
Value is PRL_UINT32 (p. 798) Look also at PLRK_RUNNING_VMS_LIMIT

_PRL_LOGIN_LOCAL_FLAGS Enumeration

PrlSrv_LoginLocal (p. 276) flags

Syntax

```
enum _PRL_LOGIN_LOCAL_FLAGS {
  PLLF_LOGIN_WITH_DELAYED_CONNECTION = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

_PRL_MASS_STORAGE_INTERFACE_TYPE Enumeration

Mass storage device interface type

Syntax

```
enum _PRL_MASS_STORAGE_INTERFACE_TYPE {
  PMS_IDE_DEVICE = 0,
  PMS_SCSI_DEVICE = 1,
  PMS_SATA_DEVICE = 2,
  PMS_VIRTIO_BLOCK_DEVICE = 3,
  PMS_UNKNOWN_DEVICE = 255
};
```

File

PrlEnums.h

_PRL_MEMGUARANTEE_TYPE Enumeration

Syntax

```
enum _PRL_MEMGUARANTEE_TYPE {
  PRL_MEMGUARANTEE_AUTO = 0,
  PRL_MEMGUARANTEE_PERCENTS = 1
};
```

File

PrlEnums.h

_PRL_MESSAGE_TYPE_ENUM Enumeration

Message type enumeration.

Syntax

```
enum _PRL_MESSAGE_TYPE_ENUM {
  PMT_WARNING = 0,
  PMT_INFORMATION = 1,
  PMT_CRITICAL = 2,
  PMT_QUESTION = 3,
  PMT_ANSWER = 4
};
```

File

PrlEnums.h

_PRL_MOBILE_ADVANCED_AUTH_MODE Enumeration

Authorization mode, when mobile client establishing remote access to the host.

Syntax

```
enum _PRL_MOBILE_ADVANCED_AUTH_MODE {
  PMAA_NO_ADVANCED_AUTH_NEEDED = 0,
  PMAA_USE_SYSTEM_CREDENTIALS = 1
};
```

File

PrlEnums.h

_PRL_MOUNT_VIRTUAL_DISK Enumeration

Mount virtual disk flags

Syntax

```
enum _PRL_MOUNT_VIRTUAL_DISK {
  PMVD_READ_ONLY = 1<<(PACF_MAX+1),
  PMVD_READ_WRITE = 1<<(PACF_MAX+2),
  PMVD_INFO = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_MOUSE_BUTTON Enumeration

Syntax

```
enum _PRL_MOUSE_BUTTON {
  PMB_NOBUTTON = 0,
  PMB_LEFT_BUTTON = 1<<0,
  PMB_RIGHT_BUTTON = 1<<1,
  PMB_MIDDLE_BUTTON = 1<<2,
  PMB_XBUTTON1 = 1<<3,
  PMB_XBUTTON2 = 1<<4,
  PMB_XBUTTON3 = 1<<5,
  PMB_XBUTTON4 = 1<<6,
  PMB_XBUTTON5 = 1<<7
};
```

File

PrlEnums.h

_PRL_NET_ADAPTER_EMULATED_TYPE Enumeration

Network adapter emulated type

Syntax

```
enum _PRL_NET_ADAPTER_EMULATED_TYPE {
  PNA_HOST_ONLY = 0,
  PNA_SHARED = 1,
  PNA_BRIDGED_NETWORK = 2,
  PNA_BRIDGED_ETHERNET = PNA_BRIDGED_NETWORK,
  PNA_DIRECT_ASSIGN = 4,
  PNA_ROUTED = 5,
  PNA_BRIDGE = 6
};
```

File

PrlEnums.h

_PRL_NET_ADAPTER_TYPE Enumeration

Syntax

```
enum _PRL_NET_ADAPTER_TYPE {
  NIC_TYPE_PHYSICAL = 0,
  NIC_TYPE_BRIDGE = 1,
  NIC_TYPE_VLAN = 2,
  NIC_TYPE_VETH = 3
};
```

File

PrlEnums.h

_PRL_NET_VIRTUAL_NETWORK_TYPE Enumeration

Virtual Network type.

Syntax

```
enum _PRL_NET_VIRTUAL_NETWORK_TYPE {
  PVN_BRIDGED_ETHERNET = 0,
  PVN_HOST_ONLY = 1
};
```

File

PrlEnums.h

Remarks

Should match values in LibrariesPrlNetworkingprl_netconfig.h

_PRL_NETFILTER_MODE Enumeration

Netfilter modes list

Syntax

```
enum _PRL_NETFILTER_MODE {
  PCNM_NOT_SET = 0,
  PCNM_DISABLED = 1,
  PCNM_STATELESS = 2,
  PCNM_STATEFUL = 3,
  PCNM_FULL = 4
};
```

File

PrlEnums.h

_PRL_NETWORK_RATE_MPU Enumeration

Special values of network packet MPU for packet rate limitation.

Syntax

```
enum _PRL_NETWORK_RATE_MPU {
  NRM_DISABLED = -1,
  NRM_ENABLED = 0
};
```

File

PrlEnums.h

Members

NRM_ENABLED
default MPU

_PRL_ONLINE_COMPACT_MODE Enumeration

Online compact mode for virtual machine disks

Syntax

```
enum _PRL_ONLINE_COMPACT_MODE {
  POCM_DISABLED = 0,
  POCM_ENABLED = 1,
  POCM_AUTO = 2
};
```

File

PrlEnums.h

_PRL_OPTIMIZE_MODIFIERS_MODE Enumeration

Determines the mode of modifiers handling inside the guest

Syntax

```
enum _PRL_OPTIMIZE_MODIFIERS_MODE {
  POD_STANDARD = 0,
  POD_OPTIMIZE_FOR_GAMES = 1,
  POD_OPTIMIZE_FOR_ACCESSIBILITY = 2
};
```

File

PrlEnums.h

_PRL_PARAM_FIELD_DATA_TYPE Enumeration

Event parameters fields types

Syntax

```
enum _PRL_PARAM_FIELD_DATA_TYPE {
  PFD_UINT32 = 0,
  PFD_STRING = 1,
  PFD_INT32 = 2,
  PFD_CDATA = 3,
  PFD_BOOLEAN = 4,
  PFD_ENTITY = 5,
  PFD_ENUMERATION = 6,
  PFD_INT64 = 7,
  PFD_UINT64 = 8,
  PFD_BINARY = 9,
  PFD_UNKNOWN = 0xFF
};
```

File

PrlEnums.h

_PRL_PORT_FORWARDING_TYPE Enumeration

Port forwarding type: PPF_TCP : TCP port forwarding PPF_UDP : UDP port forwarding

Syntax

```
enum _PRL_PORT_FORWARDING_TYPE {
  PPF_TCP = 0,
  PPF_UDP = 1
};
```

File

PrlEnums.h

_PRL_PRINTER_INTERFACE_TYPE Enumeration

Printer device interface type

Syntax

```
enum _PRL_PRINTER_INTERFACE_TYPE {
  PRN_LPT_DEVICE = 0,
  PRN_USB_DEVICE = 1,
  PRN_UNKNOWN_DEVICE = 255
};
```

File

PrlEnums.h

_PRL_PRIVATE_EVENT_TYPE Enumeration

Private event types.

Syntax

```
enum _PRL_PRIVATE_EVENT_TYPE {
  PET_IO_DISPLAY_PALETTE_CHANGED = 200000,
  PET_IO_TOOLS_TIS_QUERY = 200001,
  PET_IO_SCREEN_UPDATED = 200002,
  PET_IO_AUTH_RESPONSE = 200003,
  PET_IO_ATTACH_RESPONSE = 200004,
  PET_IO_ENCODING_RESPONSE = 200005,
  PET_IO_MEMORY_INFO = 200006,
  PET_IO_SCREEN_CAPTURED_BUFFER = 200007,
  PET_IO_TOOLS_TIS_SUBSCRIBE = 200008,
  PET_IO_TOOLS_TIS_UNSUBSCRIBE = 200009,
  PET_IO_CLI_AUTHENTICATE_SESSION = 200100,
  PET_IO_CLI_ATTACH_TO_VM = 200101,
  PET_IO_CLI_ENCODING_REQUEST = 200102,
  PET_IO_CLI_SCREEN_WATCH_UPDATES = 200103,
  PET_IO_CLI_SCREEN_CAPTURE_BUFFER = 200104,
  PET_IO_CLI_SCREEN_SIZE = 200105,
  PET_IO_CLI_KEYBOARD_SCANCODE = 200106,
  PET_IO_CLI_MOUSE_SETPOS = 200107,
```

```
    PET_IO_CLI_MOUSE_MOVE = 200108,
    PET_IO_CLI_MOUSE_NEED_CURSOR = 200109,
    PET_IO_CLI_GRACEFUL_SHUTDOWN = 200110,
    PET_IO_CLI_SCREEN_SET_SURFACE = 200111,
    PET_IO_CLI_REMOTE_COMMAND = 200112,
    PET_IO_CLI_TOOLS_UTILITY_COMMAND = 200113,
    PET_IO_CLI_TOOLS_TIS_QUERY = 200114,
    PET_IO_CLI_TOOLS_CLIPBOARD_DATA = 200115,
    PET_IO_CLI_TOOLS_SIA_DATA = 200116,
    PET_IO_CLI_TOOLS_VMCTG_COMMAND = 200117,
    PET_IO_CLI_TOOLS_GENERAL_COMMAND = 200118,
    PET_IO_CLI_TOOLS_EMPTY_RECYCLE_BIN = 200119,
    PET_IO_CLI_TOOLS_SET_TASKBAR_VISIBILITY = 200120,
    PET_IO_CLI_TOOLS_TOGGLE_DESKTOP = 200121,
    PET_IO_CLI_TOOLS_MOVE_TASKBAR = 200122,
    PET_IO_CLI_TOOLS_SHOW_RECYCLE_BIN = 200123,
    PET_IO_CLI_TOOLS_NOTIFY_COHERENCE_STATE = 200124,
    PET_IO_CLI_TOOLS_ADJUST_WORKING_AREA = 200125,
    PET_IO_CLI_KEYBOARD_KEY = 200126,
    PET_IO_CLI_TOOLS_UIEMU = 200127,
    DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_STREAM_START = 200128,
    DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_STREAM_STOP = 200129,
    DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_SET_ENCODING = 200130,
    PET_IO_CLI_CVSRC_CONNECT = 200131,
    PET_IO_CLI_CVSRC_DISCONNECT = 200132,
    PET_IO_CLI_AUTHENTICATE_EXEC_SESSION = 200133,
    PET_IO_CLI_SWITCH_SLIDING_MOUSE = 200134,
    PET_IO_CLI_SSO_SET_CREDENTIALS = 200135,
    PET_IO_CLI_SSO_SET_CREDENTIALS_ASYNC = 200136,
    PET_IO_CLI_SSO_LOCKED = 200137,
    PET_IO_CLI_SSO_UNLOCKED = 200138,
    PET_IO_CLI_DPI_SCREEN_SIZE = 200139,
    PET_IO_CLI_TOOLS_SET_INDENTS = 200140,
    PET_IO_CLI_AUDIO_RECORDING_STREAM_START = 200141,
    PET_IO_CLI_AUDIO_RECORDING_STREAM_STOP = 200142,
    PET_IO_CLI_AUDIO_PLAYOUT_STREAM_START = 200143,
    PET_IO_CLI_AUDIO_PLAYOUT_STREAM_STOP = 200144,
    PET_IO_CLI_TOOLS_OPEN_FILE_BROWSER = 200145,
    PET_IO_STDIN_PORTION = 200200,
    PET_IO_STDOUT_PORTION = 200201,
    PET_IO_STDERR_PORTION = 200202,
    PET_IO_READY_TO_ACCEPT_STDOUT_STDERR_PKGS = 200203,
    PET_IO_FIN_TO_TRANSMIT_STDOUT_STDERR = 200204,
    PET_IO_CLIENT_PROCESSED_ALL_DESCS_DATA = 200205,
    PET_IO_TTY_SETTINGS_REQUIRED = 200206,
    PET_IO_TTY_SETTINGS_PROVIDED = 200207,
    PET_IO_STDIN_WAS_CLOSED = 200208,
    PET_IT_RECONFIG_BOOTCAMP_START = 200300,
    PET_IT_VALIDATE_BOOTCAMP_START = 200301,
    PET_IT_CONVERT_TO_OLD_START = 200302,
    PET_IT_CONVERT_TO_CURRENT_START = 200303,
    PET_IT_RECONFIG_START = 200304,
    PET_IT_MERGE_SNAPSHOTS_START = 200305,
    PET_IT_MAKE_PLAIN_START = 200306,
    PET_IT_MAKE_EXPANDING_START = 200307,
    PET_IT_MAKE_SPLIT_START = 200308,
    PET_IT_MAKE_NON_SPLIT_START = 200309,
    PET_IT_GET_RESIZE_INFORMATION_START = 200310,
    PET_IT_INCREASE_LAST_VOLUME_START = 200311,
    PET_IT_RESIZING_START = 200312,
    PET_IT_DECREASING_START = 200313,
    PET_IT_COMPACTING_START = 200314,
    PET_IT_FIX_CONSISTENCY_START = 200315,
    PET_IT_CONVERT_TO_EXTEND_START = 200316,
    PET_DESKCTL_SESSION_AUTH = 200400,
    PET_DESKCTL_SESSION_AUTH_FIN = 200401,
```

```
    PET_DESKCTL_ATTACH = 200402,
    PET_DESKCTL_ATTACH_FIN = 200403,
    PET_DESKCTL_LOCAL_AUTH = 200404,
    PET_DESKCTL_LOCAL_AUTH_QUEST = 200405,
    PET_DESKCTL_LOCAL_AUTH_ANSWER = 200406,
    PET_DESKCTL_LOCAL_AUTH_FIN = 200407,
    PET_DESKCTL_ACCEPT_CLIENTS = 200408,
    PET_DESKCTL_REJECT_CLIENTS = 200409,
    PET_DESKCTL_CONFIG = 200410,
    PET_DESKCTL_CONFIG_QUERY = 200411,
    PET_DESKCTL_SEND_SAS = 200412,
    PET_DESKCTL_STATUS_CHANGE = 200413,
    PET_IO_DISCONNECT_WITH_REASON = 200500,
    PET_IO_DISPLAY_RESOLUTION_SET = 200510,
    PET_VIDEO_STREAM_START = 200600,
    PET_VIDEO_STREAM_CLOSE = 200601,
    PET_VIDEO_STREAM_FORMAT = 200602,
    PET_VIDEO_STREAM_DATA = 200603,
    PET_VIDEO_STREAM_CLOSED = 200604,
    PET_VIDEO_STREAM_ACK_FRAME = 200605,
    PET_VIDEO_STREAM_OPEN = 200606,
    PET_VIDEO_DATA_CHANNEL_CTL = 200607,
    PET_VIDEO_STREAM_STOP = 200608,
    PET_IO_PTM_PROTOCOL = 200700,
    PET_IO_IOSERVICE_CHANNELS_PROTOCOL = 200800
};
```

File

PrlEventsValues.h

Members

PET_IO_DISPLAY_PALETTE_CHANGED
BASE VALUE SET TO 200000

PET_IO_CLI_AUTHENTICATE_SESSION
ALIGNED VALUE TO 200100

PET_IO_STDIN_PORTION
ALIGNED VALUE TO 200200

PET_IT_RECONFIG_BOOTCAMP_START
ALIGNED VALUE TO 200300

PET_DESKCTL_SESSION_AUTH
ALIGNED VALUE TO 200400

PET_IO_DISCONNECT_WITH_REASON
ALIGNED VALUE TO 200500

PET_IO_DISPLAY_RESOLUTION_SET
ALIGNED VALUE TO 200510

PET_VIDEO_STREAM_START
ALIGNED VALUE TO 200600

PET_IO_PTM_PROTOCOL
ALIGNED VALUE TO 200700

PET_IO_IOSERVICE_CHANNELS_PROTOCOL
ALIGNED VALUE TO 200800

Remarks

Is NOT a part of public API.

_PRL_PROBLEM_REPORT_FLAGS Enumeration

Virtuozzo problem report flags functionality: PPRF_ADD_CLIENT_PART : specifies to add client information to problem report PPRF_ADD_SERVER_PART : specifies to add server information to problem report PPRF_DO_NOT_CREATE_HOST_SCREENSHOT : specifies do not create and add host screenshot to generating problem report

Syntax

```
enum _PRL_PROBLEM_REPORT_FLAGS {
  PPRF_ADD_CLIENT_PART = 1<<(PACF_MAX+1),
  PPRF_ADD_SERVER_PART = 1<<(PACF_MAX+2),
  PPRF_DO_NOT_CREATE_HOST_SCREENSHOT = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_PROBLEM_REPORT_REASON Enumeration

Problem report reason enumeration

Syntax

```
enum _PRL_PROBLEM_REPORT_REASON {
  PRR_UNKNOWN = 0,
  PRR_BOOT_CAMP = 1,
  PRR_HANG_LOCKUP = 2,
  PRR_KEYBOARD = 3,
  PRR_NETWORK = 4,
  PRR_PARALLELS_TOOLS = 5,
  PRR_PERFORMANCE = 6,
  PRR_PRODUCT_REGISTRATION = 7,
  PRR_SNAPSHOTS = 8,
  PRR_SOUND = 9,
  PRR_SUPPORT_REQUEST = 10,
  PRR_USB = 11,
  PRR_VIDEO_3D_GRAPHICS = 12,
  PRR_OTHER = 13,
  PRR_BACKUP = 14,
  PRR_MIGRATION = 15,
  PRR_CONNECTION = 16,
  PRR_LAST = PRR_CONNECTION
};
```

File

PrlEnums.h

Members

PRR_LAST
Should equals to the last item

_PRL_PROBLEM_REPORT_SCHEME Enumeration

Problem report scheme enumaration

Syntax

```
enum _PRL_PROBLEM_REPORT_SCHEME {
  PRS_OLD_XML_BASED = 0,
  PRS_NEW_PACKED = 1
};
```

File

PrlEnums.h

_PRL_PROBLEM_REPORT_TYPE Enumeration

Problem report type enumeration

Syntax

```
enum _PRL_PROBLEM_REPORT_TYPE {
  PRT_REPORT_TYPE_UNDEFINED = 0,
  PRT_AUTOMATIC_VM_GENERATED_REPORT = 1,
  PRT_AUTOMATIC_DISPATCHER_GENERATED_REPORT = 2,
  PRT_USER_DEFINED_ON_STOPPED_VM_REPORT = 3,
  PRT_USER_DEFINED_ON_RUNNING_VM_REPORT = 4,
  PRT_USER_DEFINED_ON_CONNECTED_SERVER = 5,
  PRT_USER_DEFINED_ON_DISCONNECTED_SERVER = 6,
  PRT_USER_DEFINED_ON_NOT_RESPONDING_VM_REPORT = 7,
  PRT_AUTOMATIC_DETECTED_REPORT = 8,
  PRT_AUTOMATIC_STATISTICS_REPORT = 9,
  PRT_AUTOMATIC_INSTALLED_SOFTWARE_REPORT = 10,
  PRT_AUTOMATIC_VZ_STATISTICS_REPORT = 11,
  PRT_AUTOMATIC_UTW7AGENT_PRE_MIGRATION = 12,
  PRT_AUTOMATIC_UTW7AGENT_POST_MIGRATION = 13,
  PRT_AUTOMATIC_TRANSPORTER_REPORT = 14,
  PRT_USER_DEFINED_TRANSPORTER_REPORT = 15,
  PRT_AUTOMATIC_GUEST_GENERATED_REPORT = 16,
  PRT_USER_DEFINED_ON_CONTAINER_REPORT = 17,
  PRT_USER_DEFINED_ON_MOBILE_REPORT = 18,
  PRT_USER_DEFINED_ON_MOBILE_FEEDBACK_REPORT = 19,
  PRT_LAST = PRT_USER_DEFINED_ON_MOBILE_FEEDBACK_REPORT
};
```

File

PrlEnums.h

Members

PRT_LAST
Should equals to the last item

_PRL_PROCESS_STATE_TYPE Enumeration

Process state element

Syntax

```
enum _PRL_PROCESS_STATE_TYPE {
  PPS_PROC_SLEEP = 0,
  PPS_PROC_RUN = 1,
  PPS_PROC_STOP = 2,
  PPS_PROC_ZOMBIE = 3,
  PPS_PROC_IDLE = 4
};
```

File

PrlEnums.h

_PRL_PRODUCT_ENUM Enumeration

Virtuozzo license products enumeration.

Syntax

```
enum _PRL_PRODUCT_ENUM {
  PRL_PRODUCT_ANY = 0,
  PRL_PRODUCT_DESKTOP = 1,
  PRL_PRODUCT_SERVER = 2,
  PRL_PRODUCT_WORKSTATION = 3,
  PRL_PRODUCT_STM = 4,
  PRL_PRODUCT_PLAYER = 5,
  PRL_PRODUCT_DESKTOP_WL = 6
};
```

File

PrlEnums.h

_PRL_PROXY_CONNECTION_STATUS Enumeration

Proxy connection status

Syntax

```
enum _PRL_PROXY_CONNECTION_STATUS {
  PCS_DISCONNECTED = 0,
  PCS_CONNECTED = 1,
  PCS_CONNECTING = 2
};
```

File

PrlEnums.h

_PRL_QUERY_ELEMENT_FLAGS Enumeration

Query element flags

Syntax

```
enum _PRL_QUERY_ELEMENT_FLAGS {
  PRL_UIEMU_ELEMENT_SEARCH_AREA_WINDOW = 0,
  PRL_UIEMU_ELEMENT_SEARCH_AREA_GLOBAL = 1
};
```

File

PrlEnums.h

_PRL_REGISTER_3PARTY_VM_FLAGS Enumeration

PrlSrv_Register3rdPartyVm (p. 285) command flags set

Syntax

```
enum _PRL_REGISTER_3PARTY_VM_FLAGS {
  PR3F_ALLOW_UNKNOWN_OS = 1<<(PACF_MAX+1),
  PR3F_CREATE_SHADOW_VM = 1<<(PACF_MAX+2)
};
```

File

PrlCommandsFlags.h

_PRL_REGISTER_NEW_VM Enumeration

PrlVm_RegEx (p. 485) flags

Syntax

```
enum _PRL_REGISTER_NEW_VM {
  PRNVM_CREATE_FROM_LION_RECOVERY_PARTITION = PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION,
  PRNVM_ALLOW_TO_AUTO_DECREASE_HDD_SIZE = 1<<(PACF_MAX+2),
  PRNVM_PRESERVE_DISK = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_REGISTER_VM_FLAGS Enumeration

PrlSrv_RegisterVmEx (p. 288) command flags set

Syntax

```
enum _PRL_REGISTER_VM_FLAGS {
  PRVF_REGENERATE_VM_UUID = 1<<(PACF_MAX+1),
  PRVF_KEEP_OTHERS_PERMISSIONS = 1<<(PACF_MAX+2),
  PRCF_FORCE = 1<<(PACF_MAX+3),
  PRVF_REGENERATE_SRC_VM_UUID = 1<<(PACF_MAX+4),
  PRVF_IGNORE_HA_CLUSTER = 1<<(PACF_MAX+5),
  PRCF_UNREG_PRESERVE = 1<<(PACF_MAX+6)
};
```

File

PrlCommandsFlags.h

_PRL_RESIZE_IMAGE_FLAGS Enumeration

PrlVmDev_ResizeImage (p. 766) command flags set

Syntax

```
enum _PRL_RESIZE_IMAGE_FLAGS {
  PRIF_RESIZE_LAST_PARTITION = 1<<(PACF_MAX+1),
  PRIF_DISK_INFO = 1<<(PACF_MAX+2),
  PRIF_RESIZE_OFFLINE = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

_PRL_RUNNING_TASK_TYPE_ENUM Enumeration

Running task type enumeration.

Syntax

```
enum _PRL_RUNNING_TASK_TYPE_ENUM {
  RTT_RUNNING_TASK_UNKNOWN = 0,
  RTT_RUNNING_TASK_CREATE_VM = 1,
  RTT_RUNNING_TASK_REGISTER_VM = 2,
  RTT_RUNNING_TASK_CLONE_VM = 3,
  RTT_RUNNING_TASK_DELETE_VM = 4,
  RTT_RUNNING_TASK_UNREG_VM = 5,
  RTT_RUNNING_TASK_CREATE_IMAGE = 6,
  RTT_RUNNING_TASK_RESTORE_VM = 7,
  RTT_RUNNING_TASK_MIGRATE_VM = 8,
  RTT_RUNNING_TASK_COPY_IMAGE = 9
};
```

File

PrlEnums.h

_PRL_SCALE_VIEW_MODE Enumeration

Aspect ratio mode

Syntax

```
enum _PRL_SCALE_VIEW_MODE {
  PSVM_OFF = 0,
  PSVM_AUTO = 1,
  PSVM_KEEP_ASPECT_RATIO = 2,
  PSVM_KEEP_ASPECT_RATIO_BY_EXPANDING = 3,
  PSVM_IGNORE_ASPECT_RATIO = 4
};
```

File

PrlEnums.h

_PRL_SECURITY_LEVEL Enumeration

Security level enumeration.

Syntax

```
enum _PRL_SECURITY_LEVEL {
  PSL_LOW_SECURITY = 0,
  PSL_NORMAL_SECURITY = 1<<0,
  PSL_HIGH_SECURITY = 1<<1
};
```

File

PrlCommandsFlags.h

Type Parameters

PSL_LOW_SECURITY

Plain TCP/IP (no encryption).

PSL_NORMAL_SECURITY

Most important data is sent and received using SSL over TCP/IP (user credentials during login, guest OS clipboard, etc.) Other data is sent and receive using plain TCP/IP with no encryption.

PSL_HIGH_SECURITY

All of the data is sent and received using SSL.

Remarks

Defines levels of security that can be used in client/server communications.

_PRL_SERIAL_PORT_SOCKET_OPERATION_MODE Enumeration

Serial port operation mode when on socket

Syntax

```
enum _PRL_SERIAL_PORT_SOCKET_OPERATION_MODE {
  PSP_SERIAL_SOCKET_SERVER = 0,
  PSP_SERIAL_SOCKET_CLIENT = 1
};
```

File

PrlEnums.h

_PRL_SERVICE_STATUS_ENUM Enumeration

Virtuozzo Service statuses

Syntax

```
enum _PRL_SERVICE_STATUS_ENUM {
  PSS_STARTED = 0,
  PSS_STOPPED = 1,
  PSS_NOT_INSTALLED = 2,
  PSS_UNKNOWN
};
```

File

PrlEnums.h

_PRL_SET_USERPASSWD_FLAGS Enumeration

PrlVm_SetUserPasswd (p. 490) flags

Syntax

```
enum _PRL_SET_USERPASSWD_FLAGS {
  PSPF_PASSWD_CRYPTED = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Members

PSPF_PASSWD_CRYPTED
passwords are encrypted

_PRL_SOUND_CHANNEL_TYPE Enumeration

Sound channel type

Syntax

```
enum _PRL_SOUND_CHANNEL_TYPE {
  PSCT_AUTO = 0,
  PSCT_STEREO = 1,
  PSCT_SURROUND_5_1 = 2
};
```

File

PrlEnums.h

_PRL_SRV_SHUTDOWN_FLAGS Enumeration

PrlSrv_ShutdownEx (p. 298) flags

Syntax

```
enum _PRL_SRV_SHUTDOWN_FLAGS {
  PSHF_FORCE_SHUTDOWN = 1<<(PACF_MAX+1),
  PSHF_DONT_WAIT_TO_COMPLETE = 1<<(PACF_MAX+2),
  PSHF_SUSPEND_VM_TO_PRAM = 1<<(PACF_MAX+3)
};
```

File

PrlCommandsFlags.h

Members

PSHF_FORCE_SHUTDOWN
Allow shutdown with active users sessions and running vms

PSHF_DONT_WAIT_TO_COMPLETE
Send response immediately when shutdown started

PSHF_SUSPEND_VM_TO_PRAM
Use PRAM to suspend/resume VM if available

_PRL_SWITCH_SNAPSHOT_FLAGS Enumeration

PrlVm_SwitchSnapshotEx flags

Syntax

```
enum _PRL_SWITCH_SNAPSHOT_FLAGS {
  PSSF_SKIP_RESUME = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

Members

PSSF_SKIP_RESUME
skip resume stage for Vm in the

- running/paused state in the snapshot

_PRL_TASKBAR_STATES Enumeration

Taskbar states

Syntax

```
enum _PRL_TASKBAR_STATES {
  PTBS_UNKNOWN = -1,
  PTBS_HIDE = 0,
  PTBS_SHOW = 1,
  PTBS_DEFAULT = 2
};
```

File

PrlEnums.h

_PRL_TIS_RECORD_FIELD Enumeration

Virtuozzo Tools Information Service record fields

Syntax

```
enum _PRL_TIS_RECORD_FIELD {
  PTIS_RECORD_UID = 1<<0,
  PTIS_RECORD_NAME = 1<<1,
  PTIS_RECORD_INFO = 1<<2,
  PTIS_RECORD_TEXT = 1<<3,
  PTIS_RECORD_DATA = 1<<4,
  PTIS_RECORD_STATE = 1<<5,
  PTIS_RECORD_TIME = 1<<6,
  PTIS_RECORD_OWNER = 1<<7,
  PTIS_RECORD_FLAGS = 1<<8,
  PTIS_RECORD_REMOVED = 1<<30,
  PTIS_RECORD_CLEARED = 1<<31
};
```

File

PrlEnums.h

Members

PTIS_RECORD_REMOVED
set when record removed

PTIS_RECORD_CLEARED
set only with PTIS_RECORD_REMOVED bit when entire database was cleared

_PRL_TIS_RECORD_STATE Enumeration

Virtuozzo Tools Information Service record states

Syntax

```
enum _PRL_TIS_RECORD_STATE {
  PTIS_RECORD_EMPTY = 0,
  PTIS_RECORD_ACTIVE = 1,
  PTIS_RECORD_CANCELED = 2
};
```

File

PrlEnums.h

_PRL_TOOLS_SIA_OPEN_IN_TYPES Enumeration

Shared Internet Applications 'open in' type PSIA_OPEN_DEFAULT : Open host link with host app; Open guest link with guest app PSIA_OPEN_IN_GUEST : Open all links with guest apps; PSIA_OPEN_IN_HOST : Open all links with host apps;

Syntax

```
enum _PRL_TOOLS_SIA_OPEN_IN_TYPES {
  PSIA_OPEN_DEFAULT = 0,
  PSIA_OPEN_IN_GUEST = 1,
  PSIA_OPEN_IN_HOST = 2
};
```

File

PrlEnums.h

_PRL_TOOLS_UTILITY_COMMAND_FLAG Enumeration

Virtuozzo Tools Utility command flags

Syntax

```
enum _PRL_TOOLS_UTILITY_COMMAND_FLAG {
  PTU_FLG_ZERO = (1 << 0)
};
```

File

PrlCommandsFlags.h

Members

PTU_FLG_ZERO
@brief Flag is useless now, but placed here to show how and where it must be

_PRL_TOOLS_UTILITY_COMMAND_ID Enumeration

Virtuozzo Tools Utility commands identifiers

Syntax

```
enum _PRL_TOOLS_UTILITY_COMMAND_ID {
  PTU_CMD_INVALID = 0,
  PTU_CMD_PTU_ALIVE = 1
};
```

File

PrlEnums.h

Members

PTU_CMD_INVALID

brief Identifier of invalid command

PTU_CMD_PTU_ALIVE

@brief Can be sent to indicate that Virtuozzo Tools Utility subsystem is alive and ready to work

_PRL_UNDO_DISKS_MODE Enumeration

VM undo disks mode: PUD_DISABLE_UNDO_DISKS : disbale undo disks PUD_REVERSE_CHANGES : at stopping VM revert all disk changes PUD_COMMIT_CHANGES : at stopping VM apply all disk changes PUD_PROMPT_BEHAVIOUR : ask user which mode reverse or commit will be applied

Syntax

```
enum _PRL_UNDO_DISKS_MODE {
  PUD_DISABLE_UNDO_DISKS = 0,
  PUD_REVERSE_CHANGES = 1,
  PUD_COMMIT_CHANGES = 2,
  PUD_PROMPT_BEHAVIOUR = 3
};
```

File

PrlEnums.h

_PRL_UPDATE_LICENSE_FLAGS Enumeration

Update/Install license operations: Uses for PrlSrv_UpdateLicenseEx (p. 310) operation

Syntax

```
enum _PRL_UPDATE_LICENSE_FLAGS {
  PUPLF_KICK_TO_UDPATE_CURR_FILE_LICENSE = 1<<(PACF_MAX+1),
  PUPLF_ACTIVATE_PERMANENT_LICENSE_ONLINE = 1<<(PACF_MAX+2),
  PUPLF_ACTIVATE_PERMANENT_LICENSE_OFFLINE = 1<<(PACF_MAX+3),
  PUPLF_CHECK_KEY_ONLY = 1<<(PACF_MAX+4),
  PUPLF_DEACTIVATE_CURRENT_LICENSE = 1<<(PACF_MAX+5),
  PUPLF_ACTIVATE_PRECACHED_KEY = 1<<(PACF_MAX+6),
```

```
    PUPLF_STORE_PRECACHED_KEY = 1<<(PACF_MAX+7),
    PUPLF_REMOVE_PRECACHED_KEY = 1<<(PACF_MAX+8),
    PUPLF_DEACTIVATE_CURRENT_LICENSE_OFFLINE_ALLOWED = 1<<(PACF_MAX+9),
    PUPLF_CHECK_KEY_ONLY_VOLUME = 1<<(PACF_MAX+10),
    PUPLF_REGISTER_CURRENT_LICENSE = 1<<(PACF_MAX+11)
  };
```

File

PrlCommandsFlags.h

_PRL_USB_DEVICE_AUTO_CONNECT_OPTION Enumeration

USB autoconnect devices options

Syntax

```
  enum _PRL_USB_DEVICE_AUTO_CONNECT_OPTION {
    PUD_CONNECT_TO_PRIMARY_OS = 0,
    PUD_CONNECT_TO_GUEST_OS = 1,
    PUD_ASK_USER_WHAT_TODO = 2
  };
```

File

PrlEnums.h

_PRL_USB_DEVICE_CONNECT_REASON Enumeration

USB device connect/disconnect reason

Syntax

```
  enum _PRL_USB_DEVICE_CONNECT_REASON {
    PUD_CONNECTED_MANUALLY = 0,
    PUD_CONNECTED_AUTOMATICALLY = 1
  };
```

File

PrlEnums.h

_PRL_USB_DEVICE_TYPE Enumeration

USB device type

Syntax

```
  enum _PRL_USB_DEVICE_TYPE {
    PUDT_OTHER = 0,
    PUDT_HUB = 1,
    PUDT_VIDEO = 2,
    PUDT_FOTO = 3,
    PUDT_AUDIO = 4,
    PUDT_PRINTER = 5,
    PUDT_SCANNER = 6,
```

```
    PUDT_BLUETOOTH = 7,
    PUDT_WIRELESS = 8,
    PUDT_COMMUNICATION = 9,
    PUDT_KEYBOARD = 10,
    PUDT_MOUSE = 11,
    PUDT_SMART_CARD = 12,
    PUDT_DISK_STORAGE = 13,
    PUDT_ATAPI_STORAGE = 14,
    PUDT_APPLE_IPHONE = 1000,
    PUDT_APPLE_IPOD = 1001,
    PUDT_APPLE_IPAD = 1002,
    PUDT_APPLE_IETH = 1003,
    PUDT_RIM_BLACKBERRY = 2000,
    PUDT_GARMIN_GPS = 3000
};
```

File

PrlEnums.h

Members

PUDT_OTHER
unknown device type

PUDT_HUB
usb hub

PUDT_VIDEO
streaming usb video device

PUDT_FOTO
still image usb video device

PUDT_AUDIO
usb audio device

PUDT_PRINTER
usb printer

PUDT_SCANNER
usb scaner

PUDT_BLUETOOTH
usb bluetooth

PUDT_WIRELESS
usb wireless device

PUDT_COMMUNICATION
other usb communication device

PUDT_KEYBOARD
usb keyboard

PUDT_MOUSE
usb mouse

PUDT_SMART_CARD
usb smartcard reader

PUDT_DISK_STORAGE
usb storage

PUDT_ATAPI_STORAGE
usb CD/DVD-ROMs Some vendor specific usb device classes

PUDT_APPLE_IPHONE
apple iphone

PUDT_APPLE_IPOD
apple ipod

PUDT_APPLE_IPAD
apple ipad

PUDT_APPLE_IETH
apple ieth

PUDT_RIM_BLACKBERRY
rim blackberry

PUDT_GARMIN_GPS
garmin gps

_PRL_VCMMD_COMMAND_FLAGS Enumeration

Syntax

```
enum _PRL_VCMMD_COMMAND_FLAGS {
  VCMMD_COMMAND_RUNTIME = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

_PRL_VERSION_ENUM Enumeration

Virtuozzo license versions enumeration.

Syntax

```
enum _PRL_VERSION_ENUM {
  PRL_VERSION_ANY = 0,
  PRL_VERSION_3X = 1,
  PRL_VERSION_4X = 2,
  PRL_VERSION_5X = 3,
  PRL_VERSION_6X = 4,
  PRL_VERSION_7X = 5,
  PRL_VERSION_8X = 6,
  PRL_VERSION_9X = 7,
  PRL_VERSION_10X = 8
};
```

File

PrlEnums.h

_PRL_VIDEO_3D_ACCELERATION Enumeration

Video 3D acceleration

Syntax

```
enum _PRL_VIDEO_3D_ACCELERATION {
  P3D_DISABLED = 0,
  P3D_ENABLED_HIGHEST = 1,
  P3D_ENABLED_DX9 = 2
};
```

File

PrlEnums.h

_PRL_VIRTUAL_MACHINE_PROFILE Enumeration

Virtual machine profiles enum

Syntax

```
enum _PRL_VIRTUAL_MACHINE_PROFILE {
  VMP_PRODUCTIVITY = 0,
  VMP_GAMES = 1,
  VMP_SOFTWARE_DEVELOPER = 2,
  VMP_DESIGNER = 3
};
```

File

PrlEnums.h

_PRL_VIRTUAL_NETWORK_FLAGS Enumeration

PrlSrv_AddVirtualNetwork (p. 224) flangs

Syntax

```
enum _PRL_VIRTUAL_NETWORK_FLAGS {
  PRL_USE_VNET_NAME_FOR_BRIDGE_NAME = 1<<(PACF_MAX+1)
};
```

File

PrlCommandsFlags.h

_PRL_VM_ON_CRASH_ACTION Enumeration

Specifies an on-crash actions

Syntax

```
enum _PRL_VM_ON_CRASH_ACTION {
  POCA_PAUSE = 0,
  POCA_RESTART = 1
};
```

File

PrlEnums.h

_PRL_VM_ON_CRASH_OPTION Enumeration

Specifies an on-crash options

Syntax

```
enum _PRL_VM_ON_CRASH_OPTION {
  POCO_DUMP = 1<<(PACF_MAX+1),
  POCO_REPORT = 1<<(PACF_MAX+2),
  POCO_FULL = POCO_DUMP|POCO_REPORT
};
```

File

PrlCommandsFlags.h

_PRL_VMCTG_EVENT Enumeration

Tools events, passed to the callback (see PRL_VMCTG_TOOL_DESC and PRL_TOOL_HANDLER_PTR above)

Syntax

```
enum _PRL_VMCTG_EVENT {
  TEV_DATA_RECEIVED,
  TEV_CONNECTED,
  TEV_DISCONNECTED
};
```

File

PrlEnums.h

Members

TEV_DATA_RECEIVED
: pData is PRL_TOOL_DATA_RECEIVED

TEV_CONNECTED
: pData is null

TEV_DISCONNECTED
: pData is null

_PRL_VTD_STATUS_CODE Enumeration

VTD status code

Syntax

```
enum _PRL_VTD_STATUS_CODE {
  PVSC_DMAR_NOT_FOUND = -5,
  PVSC_DMAR_ERROR = -4,
  PVSC_ACPI_ERROR = -3,
  PVSC_HARDWARE_ERROR = -2,
  PVSC_INTERNAL_ERROR = -1,
  PVSC_NOT_PRESENT = 0,
  PVSC_ENABLED = 1
};
```

File

PrlEnums.h

_PRL_WINDOW_CORNERS Enumeration

Window corners

Syntax

```
enum _PRL_WINDOW_CORNERS {
  PWC_TOP_LEFT_CORNER = 0,
  PWC_TOP_RIGHT_CORNER = 1,
  PWC_BOTTOM_LEFT_CORNER = 2,
  PWC_BOTTOM_RIGHT_CORNER = 3
};
```

File

PrlEnums.h

_PRL_WINDOW_EDGE_MASK Enumeration

Edges where taskbar can be placed (in coherence mode).

Syntax

```
enum _PRL_WINDOW_EDGE_MASK {
  WINDOW_LEFT_EDGE_MASK = (0x1L << 0),
  WINDOW_TOP_EDGE_MASK = (0x1L << 1),
  WINDOW_RIGHT_EDGE_MASK = (0x1L << 2),
  WINDOW_BOTTOM_EDGE_MASK = (0x1L << 3),
  WINDOW_ALL_EDGE_MASK =
WINDOW_LEFT_EDGE_MASK|WINDOW_TOP_EDGE_MASK|WINDOW_RIGHT_EDGE_MASK|WINDOW_BOTTOM_EDGE_MASK
};
```

File

PrlEnums.h

_PrlPlugin Structure

Base interface

Syntax

```
struct _PrlPlugin {
  void (* Release)(struct _PrlPlugin* _self);
};
```

File

PrlPluginClasses.h

PRL_ADAPTER_INDEX Enumeration

Virtuozzo adapter index reserved values: PAI_INVALID_ADAPTER: Virtuozzo adapter in general is not available PAI_GENERATE_INDEX: calculate a new Virtuozzo adapter index

Syntax

```
typedef enum _PRL_ADAPTER_INDEX {
  PAI_INVALID_ADAPTER = -1,
  PAI_GENERATE_INDEX = -2
} PRL_ADAPTER_INDEX;
```

File

PrlEnums.h

PRL_ALLOWED_VM_COMMAND Enumeration

Allowed vm commands enumeration

Syntax

```
typedef enum _PRL_ALLOWED_VM_COMMAND {
  PAR_VM_START_ACCESS = 0,
  PAR_VM_STOP_ACCESS = 1,
  PAR_VM_PAUSE_ACCESS = 2,
  PAR_VM_RESET_ACCESS = 3,
  PAR_VM_SUSPEND_ACCESS = 4,
  PAR_VM_RESUME_ACCESS = 5,
  PAR_VM_DROPSUSPENDEDSTATE_ACCESS = 6,
  PAR_VM_CLONE_ACCESS = 7,
  PAR_VM_DELETE_ACCESS = 8,
  PAR_VM_GETPROBLEMREPORT_ACCESS = 9,
  PAR_VM_GETCONFIG_ACCESS = 10,
  PAR_VM_GETSTATISTICS_ACCESS = 11,
  PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS = 12,
  PAR_VM_SUBSCRIBETOGUESTSTATISTICS_ACCESS = PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS,
  PAR_VM_UNSUBSCRIBEFROMGUESTSTATISTICS_ACCESS = PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS,
  PAR_VM_UNREG_ACCESS = 13,
  PAR_VM_COMMIT_ACCESS = 14,
  PAR_VM_EDITING_ACCESS = PAR_VM_COMMIT_ACCESS,
```

```
    PAR_VMDEV_CONNECT_ACCESS = 15,
    PAR_VMDEV_DISCONNECT_ACCESS = 16,
    PAR_VMDEV_CREATEIMAGE_ACCESS = 17,
    PAR_VM_SEND_ANSWER_ACCESS = 18,
    PAR_VM_GET_VMINFO_ACCESS = 19,
    PAR_VM_INSTALL_TOOLS_ACCESS = 20,
    PAR_VM_INITIATE_DEV_STATE_NOTIFICATIONS_ACCESS = 21,
    PAR_VM_UPDATE_SECURITY_ACCESS = 22,
    PAR_VM_CANCEL_COMPACT_ACCESS = 23,
    PAR_VM_REGISTER_ACCESS = 24,
    PAR_VM_MIGRATE_ACCESS = 25,
    PAR_VM_START_STOP_VNC_SERVER = 26,
    PAR_VM_GET_SUSPENDED_SCREEN_ACCESS = 27,
    PAR_VM_PERFSTAT_ACCESS = 28,
    PAR_VM_LOCK_ACCESS = 29,
    PAR_VM_MIGRATE_CANCEL_ACCESS = PAR_VM_MIGRATE_ACCESS,
    PAR_VM_RUN_COMPRESSOR_ACCESS = 31,
    PAR_VM_CANCEL_COMPRESSOR_ACCESS = 32,
    PAR_VM_CREATE_SNAPSHOT_ACCESS = 33,
    PAR_VM_SWITCH_TO_SNAPSHOT_ACCESS = 34,
    PAR_VM_DELETE_SNAPSHOT_ACCESS = 35,
    PAR_VM_START_EX_ACCESS = 36,
    PAR_VM_UNLOCK_ACCESS = 37,
    PAR_VM_RESTART_GUEST_ACCESS = 38,
    PAR_VM_INSTALL_UTILITY_ACCESS = 39,
    PAR_VM_UPDATE_TOOLS_SECTION = 40,
    PAR_VM_RESTORE_ACCESS = 41,
    PAR_VM_CREATE_BACKUP_ACCESS = 42,
    PAR_VM_RESIZE_DISK_ACCESS = 43,
    PAR_VM_CREATE_ACCESS = 44,
    PAR_SRV_USER_PROFILE_BEGINEDIT_ACCESS = 45,
    PAR_SRV_USER_PROFILE_COMMIT_ACCESS = 46,
    PAR_SRV_SERVER_PROFILE_BEGINEDIT_ACCESS = 47,
    PAR_SRV_SERVER_PROFILE_COMMIT_ACCESS = 48,
    PAR_VM_GUI_VIEW_MODE_CHANGE_ACCESS = 49,
    PAR_VM_BEGINEDIT_ACCESS = 50,
    PAR_VM_RESTORE_BACKUP_ACCESS = 51,
    PAR_VM_COMPACT_ACCESS = 52,
    PAR_VM_CHANGE_SID_ACCESS = 53,
    PAR_VM_RESET_UPTIME_ACCESS = 54,
    PAR_VM_INTERNAL_CMD_ACCESS = 55,
    PAR_VM_GET_VIRT_DEV_INFO = 56,
    PAR_VM_GET_TOOLS_INFO = 57,
    PAR_VM_CONVERT_DISKS_ACCESS = 58,
    PAR_VM_ENCRYPT_OPERATION_ACCESS = 59,
    PAR_VM_CHANGE_PASSWORD_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_ENCRYPT_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_DECRYPT_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_SET_PROTECTION_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_REMOVE_PROTECTION_ACCESS = PAR_VM_ENCRYPT_OPERATION_ACCESS,
    PAR_VM_AUTHORISE_ACCESS = 60,
    PAR_VMDEV_COPY_IMAGE_ACCESS = 61,
    PAR_VM_MOUNT_ACCESS = 62,
    PAR_VM_MOVE_ACCESS = 63,
    PAR_VM_CHANGE_GUEST_OS_PASSWORD_ACCESS = 64,
    PAR_SRV_SENDPROBLEMREPORT_ACCESS = 65,
    PAR_MAX
  } PRL_ALLOWED_VM_COMMAND;
```

File

PrlEnums.h

Members

PAR_MAX

should be the last

PRL_API_COMMAND_FLAGS Enumeration

Common operations flags set.

Syntax

```
typedef enum _PRL_API_COMMAND_FLAGS {
  PACF_NORMAL_SECURITY = PSL_NORMAL_SECURITY,
  PACF_HIGH_SECURITY = PSL_HIGH_SECURITY,
  PACF_NON_INTERACTIVE_MODE = 1<<2,
  PACF_CANCEL_TASK_ON_END_SESSION = 1<<3,
  PACF_MAX = 10,
  PACF_LAST = 1<<PACF_MAX
} PRL_API_COMMAND_FLAGS;
```

File

PrlCommandsFlags.h

Members

PACF_NORMAL_SECURITY
Client connection security mode

PACF_NON_INTERACTIVE_MODE
Session interactive mode

PACF_CANCEL_TASK_ON_END_SESSION
Cancel long-term operation by session disconnect

PACF_MAX
Interval from here to PACF_LAST reserved for further purposes.

Remarks

Please do not override values

PRL_API_INIT_FLAGS Enumeration

Virtuozzo API initialization flags: PrlApi_InitEx (p. 739) command flags set PAIF_USE_GRAPHIC_MODE : specifies create SDK session which has access to the graphic mode

Syntax

```
typedef enum _PRL_API_INIT_FLAGS {
  PAIF_USE_GRAPHIC_MODE = 1<<(PACF_MAX+1),
  PAIF_INIT_AS_APPSTORE_CLIENT = 1<<(PACF_MAX+2)
} PRL_API_INIT_FLAGS;
```

File

PrlCommandsFlags.h

PRL_APP_IN_DOCK_MODE Enumeration

VM application in Dock mode

Syntax

```
typedef enum _PRL_APP_IN_DOCK_MODE {
  PDM_APP_IN_DOCK_NEVER = 0,
  PDM_APP_IN_DOCK_COHERENCE_ONLY = 1,
  PDM_APP_IN_DOCK_ALWAYS = 2
} PRL_APP_IN_DOCK_MODE;
```

File

PrlEnums.h

PRL_APPLIANCE_DOWNLOAD_STATUS Enumeration

Appliance download status

Syntax

```
typedef enum _PRL_APPLIANCE_DOWNLOAD_STATUS {
  PADS_NOT_STARTED = 1,
  PADS_STOPPED = 2,
  PADS_DOWNLOADING = 3,
  PADS_DOWNLOADED = 4,
  PADS_UNPACKED = 5,
  PADS_MOVED = 6,
  PADS_REGISTERED = 7,
  PADS_CANCELED = 8
} PRL_APPLIANCE_DOWNLOAD_STATUS;
```

File

PrlEnums.h

PRL_APPLICATION_MODE Enumeration

Virtuozzo application mode in which SDK should work PAM_SERVER : SDK should working at Virtuozzo Service application mode

Syntax

```
typedef enum _PRL_APPLICATION_MODE {
  PAM_UNKNOWN = 0xffff,
  PAM_SERVER = 0,
  PAM_LAST = PAM_SERVER
} PRL_APPLICATION_MODE;
```

File

PrlEnums.h

Members

PAM_LAST

PAM_LAST should be always last entry in enum !

PRL_BEGIN_VM_BACKUP_FLAGS Enumeration

PrlVm_BeginBackup flags.

Syntax

```
typedef enum _PRL_BEGIN_VM_BACKUP_FLAGS {
  PBMBF_CREATE_MAP = 1<<(PACF_MAX+1)
} PRL_BEGIN_VM_BACKUP_FLAGS;
```

File

PrlCommandsFlags.h

Members

PBMBF_CREATE_MAP

Create a disk map

PRL_BIOS_TYPE Enumeration

BIOS type

Syntax

```
typedef enum _PRL_BIOS_TYPE {
  PBT_LEGACY = 0,
  PBT_EFI = 1
} PRL_BIOS_TYPE;
```

File

PrlEnums.h

PRL_CHS Structure

CHS format of disk

Syntax

```
typedef struct _PRL_CHS {
  PRL_UINT32 Sectors;
  PRL_UINT32 Heads;
  PRL_UINT32 Cylinders;
} PRL_CHS;
```

File

PrlDisk.h

PRL_CLONE_VM_FLAGS Enumeration

PrlVm_CloneEx (p. 451) command flags set

PCVF_DETACH_EXTERNAL_VIRTUAL_HDD

- to remove link to Virtual Hdd which located outside of VM.pvm directory from cloned vm configuration.

Syntax

```
typedef enum _PRL_CLONE_VM_FLAGS {
  PCVF_CLONE_TO_TEMPLATE = 1<<(PACF_MAX+1),
  PCVF_CHANGE_SID = 1<<(PACF_MAX+2),
  PCVF_LINKED_CLONE = 1<<(PACF_MAX+3),
  PCVF_IMPORT_BOOT_CAMP = 1<<(PACF_MAX+4),
  PCVF_DETACH_EXTERNAL_VIRTUAL_HDD = 1<<(PACF_MAX+5)
} PRL_CLONE_VM_FLAGS;
```

File

PrlCommandsFlags.h

PRL_CLUSTERED_DEVICE_SUBTYPE Enumeration

Clustered device subtype

Syntax

```
typedef enum _PRL_CLUSTERED_DEVICE_SUBTYPE {
  PCD_BUSLOGIC = 0,
  PCD_LSI_SPI = 1,
  PCD_LSI_SAS = 2,
  PCD_VIRTIO_SCSI = 3,
  PCD_UNKNOWN_DEVICE = 255
} PRL_CLUSTERED_DEVICE_SUBTYPE;
```

File

PrlEnums.h

PRL_COMMIT_ENCRYPTION_FLAGS Enumeration

PrlVm_CommitEncryption (p. 454) command flags

Syntax

```
typedef enum _PRL_COMMIT_ENCRYPTION_FLAGS {
  PCEF_CHANGE_MASTER_KEY = 1<<(PACF_MAX+1),
  PCEF_DONT_SHRED_PLAIN_DATA = 1<<(PACF_MAX+2)
} PRL_COMMIT_ENCRYPTION_FLAGS;
```

File

PrlCommandsFlags.h

Members

PCEF_CHANGE_MASTER_KEY

Normally, only the master key (MK is used to actually encrypt the data on a virtual disk) is being re-encrypted when the disk encryption key identifier is changed for an already encrypted disk.

PCEF_DONT_SHRED_PLAIN_DATA

When a plain disk is being encrypted, the encrypted copy of the original disk is created.

PRL_COMPACT_MODE Enumeration

Compact mode: PCM_COMPACT_WITH_HARD_DISKS_INFO : show question before compacting with size disks info PCM_HARD_DISKS_INFO : just get disk info PCM_FULL_CLEAN_UP_VM : compact all disks and delete all unneccessary files

Syntax

```
typedef enum _PRL_COMPACT_MODE {
  PCM_COMPACT_WITH_HARD_DISKS_INFO = 1<<(PACF_MAX+1),
  PCM_HARD_DISKS_INFO = 1<<(PACF_MAX+2),
  PCM_FULL_CLEAN_UP_VM = 1<<(PACF_MAX+3)
} PRL_COMPACT_MODE;
```

File

PrlCommandsFlags.h

PRL_CONVERT_VIRTUAL_DISK Enumeration

Convert virtual disk

- PCVD_CANCEL : cancel disk conversion
- PCVD_TO_PLAIN_DISK : convert disk from expanding to plain disk
- PCVD_TO_EXPANDING_DISK : convert disk from plain to expanding disk
- PCVD_TO_SPLIT_DISK : convert disk from non-split to split disk
- PCVD_TO_NON_SPLIT_DISK : convert disk from split to non-split disk
- PCVD_MERGE_ALL_SNAPSHOTS : merge all disk snapshots

Syntax

```
typedef enum _PRL_CONVERT_VIRTUAL_DISK {
  PCVD_CANCEL = 1<<(PACF_MAX+1),
  PCVD_TO_PLAIN_DISK = 1<<(PACF_MAX+2),
  PCVD_TO_EXPANDING_DISK = 1<<(PACF_MAX+3),
  PCVD_TO_SPLIT_DISK = 1<<(PACF_MAX+4),
  PCVD_TO_NON_SPLIT_DISK = 1<<(PACF_MAX+5),
  PCVD_MERGE_ALL_SNAPSHOTS = 1<<(PACF_MAX+6)
} PRL_CONVERT_VIRTUAL_DISK;
```

File

PrlCommandsFlags.h

PRL_COPY_CT_TEMPLATE_FLAGS Enumeration

copy of CT templates functionality flags set

Syntax

```
typedef enum _PRL_COPY_CT_TEMPLATE_FLAGS {
  PCTMP_LOW_SECURITY = PSL_LOW_SECURITY,
  PCTMP_NORMAL_SECURITY = PACF_NORMAL_SECURITY,
  PCTMP_HIGH_SECURITY = PACF_HIGH_SECURITY,
  PCTMPL_FORCE = 1<<(PACF_MAX+1)
} PRL_COPY_CT_TEMPLATE_FLAGS;
```

File

PrlCommandsFlags.h

Type Parameters

PCTMPL_FORCE

force template copy

PRL_CORNER_ACTIONS Enumeration

Window corner actions

Syntax

```
typedef enum _PRL_CORNER_ACTIONS {
  PCA_NO_ACTION = 0,
  PCA_WINDOWED = 1,
  PCA_COHERENCE = 2,
  PCA_SEAMLESS = 3,
  PCA_MODALITY = 4,
  PCA_HIDE_APPLICATION = 5,
  PCA_UNGRAB_INPUT = 6,
  PCA_CRYSTAL = 7
} PRL_CORNER_ACTIONS;
```

File

PrlEnums.h

PRL_CPU_FEATURES_EX Enumeration

Syntax

```
typedef enum _PRL_CPU_FEATURES_EX {
  PCFE_FEATURES = 0,
  PCFE_EXT_FEATURES = 1,
  PCFE_EXT_00000007_EBX = 2,
  PCFE_EXT_80000001_ECX = 3,
  PCFE_EXT_80000001_EDX = 4,
  PCFE_EXT_80000007_EDX = 5,
  PCFE_EXT_80000008_EAX = 6,
```

```
  PCFE_EXT_0000000D_EAX = 7,
  PCFE_MAX
} PRL_CPU_FEATURES_EX;
```

File

PrlEnums.h

Members

PCFE_MAX
have to be the last.

PRL_CPU_HVT Enumeration

CPU mode

Syntax

```
typedef enum _PRL_CPU_HVT {
  PCM_CPU_NONE_HV = 0,
  PCM_CPU_INTEL_VT_X,
  PCM_CPU_AMD_V
} PRL_CPU_HVT;
```

File

PrlEnums.h

PRL_CPU_MODE Enumeration

CPU mode

Syntax

```
typedef enum _PRL_CPU_MODE {
  PCM_CPU_MODE_32 = 0,
  PCM_CPU_MODE_64 = 1
} PRL_CPU_MODE;
```

File

PrlEnums.h

PRL_CPULIMIT_TYPE Enumeration

Syntax

```
typedef enum _PRL_CPULIMIT_TYPE {
  PRL_CPULIMIT_PERCENTS100 = 1,
  PRL_CPULIMIT_MHZ = 2,
  PRL_CPULIMIT_PERCENTS = 3,
  PRL_CPULIMIT_PERCENTS_TO_MHZ = 4
} PRL_CPULIMIT_TYPE;
```

File

PrlEnums.h

Members

PRL_CPULIMIT_PERCENTS100
cpulimit 0-100

PRL_CPULIMIT_PERCENTS
cpulimit 0-100 * nCpu

PRL_CPULIMIT_PERCENTS_TO_MHZ
The limit in % convert to Mhz

PRL_CPUS_COUNT Enumeration

constant for unlimited CPU number

Syntax

```
typedef enum _PRL_CPUS_COUNT {
  PRL_CPU_UNLIMITED = 0xffffffff
} PRL_CPUS_COUNT;
```

File

PrlEnums.h

PRL_CPUUITS Enumeration

Syntax

```
typedef enum _PRL_CPUUNITS {
  PRL_CPUUNITS_MIN = 10,
  PRL_CPUUNITS_DEFAULT = 1000,
  PRL_CPUUNITS_MAX = 500000
} PRL_CPUUITS;
```

File

PrlEnums.h

PRL_CREATE_DISK_IMAGE_FLAGS Enumeration

Create image flags

Syntax

```
typedef enum _PRL_CREATE_DISK_IMAGE_FLAGS {
  PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION = 1<<(PACF_MAX+1)
} PRL_CREATE_DISK_IMAGE_FLAGS;
```

File

PrlCommandsFlags.h

Members

PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION
create from lion recovery

PRL_CREATE_SNAPSHOT_FLAGS Enumeration

VM snapshot creation flags set

Syntax

```
typedef enum _PRL_CREATE_SNAPSHOT_FLAGS {
  PCSF_DISK_ONLY = 1<<(PACF_MAX+1),
  PCSF_BACKUP = 1<<(PACF_MAX+2)
} PRL_CREATE_SNAPSHOT_FLAGS;
```

File

PrlCommandsFlags.h

Members

PCSF_DISK_ONLY
internal flag

PCSF_BACKUP
internal flag

Type Parameters

PCSF_DISK_ONLY
to snapshot hdd's only for running Vm

PCSF_BACKUP
create snapshot for backup

PRL_CT_CAPMASK Enumeration

Capabilities list

Syntax

```
typedef enum _PRL_CT_CAPMASK {
  PCC_CHOWN = (1 << 0),
  PCC_FS_MASK = PCC_CHOWN,
  PCC_DAC_OVERRIDE = (1 << 1),
  PCC_DAC_READ_SEARCH = (1 << 2),
  PCC_FOWNER = (1 << 3),
  PCC_FSETID = (1 << 4),
  PCC_KILL = (1 << 5),
  PCC_SETGID = (1 << 6),
  PCC_SETUID = (1 << 7),
  PCC_SETPCAP = (1 << 8),
  PCC_LINUX_IMMUTABLE = (1 << 9),
  PCC_NET_BIND_SERVICE = (1 << 10),
  PCC_NET_BROADCAST = (1 << 11),
  PCC_NET_ADMIN = (1 << 12),
  PCC_NET_RAW = (1 << 13),
```

```
    PCC_IPC_LOCK = (1 << 14),
    PCC_IPC_OWNER = (1 << 15),
    PCC_SYS_MODULE = (1 << 16),
    PCC_SYS_RAWIO = (1 << 17),
    PCC_SYS_CHROOT = (1 << 18),
    PCC_SYS_PTRACE = (1 << 19),
    PCC_SYS_PACCT = (1 << 20),
    PCC_SYS_ADMIN = (1 << 21),
    PCC_SYS_BOOT = (1 << 22),
    PCC_SYS_NICE = (1 << 23),
    PCC_SYS_RESOURCE = (1 << 24),
    PCC_SYS_TIME = (1 << 25),
    PCC_SYS_TTY_CONFIG = (1 << 26),
    PCC_MKNOD = (1 << 27),
    PCC_LEASE = (1 << 28),
    PCC_AUDIT_WRITE = (1 << 29),
    PCC_VE_ADMIN = (1 << 30),
    PCC_SETFCAP = (1 << 31)
} PRL_CT_CAPMASK;
```

File

PrlEnums.h

Members

PCC_FS_MASK
FIXME: real value of PCC_FS_MASK?

PRL_CT_FEATURES Enumeration

CT Features

Syntax

```
typedef enum _PRL_CT_FEATURES {
  PCF_FEATURE_SYSFS = (1 << 0),
  PCF_FEATURE_NFS = (1 << 1),
  PCF_FEATURE_DEF_PERM = (1 << 2),
  PCF_FEATURE_SIT = (1 << 3),
  PCF_FEATURE_IPIP = (1 << 4),
  PCF_FEATURE_PPP = (1 << 5),
  PCF_FEATURE_IPGRE = (1 << 6),
  PCF_FEATURE_BRIDGE = (1 << 7),
  PCF_FEATURE_NFSD = (1 << 8)
} PRL_CT_FEATURES;
```

File

PrlEnums.h

PRL_CT_REINSTALL_OPTIONS Enumeration

Syntax

```
typedef enum _PRL_CT_REINSTALL_OPTIONS {
  REINSTALL_SKIP_BACKUP = 1<<(PACF_MAX+1),
  REINSTALL_SKIP_SCRIPTS = 2<<(PACF_MAX+1),
  REINSTALL_RESET_PWDB = 3<<(PACF_MAX+1)
} PRL_CT_REINSTALL_OPTIONS;
```

File

PrlCommandsFlags.h

PRL_CT_RESOURCE Enumeration

Syntax

```
typedef enum _PRL_CT_RESOURCE {
  PCR_KMEMSIZE = 1,
  PCR_LOCKEDPAGES = 2,
  PCR_PRIVVMPAGES = 3,
  PCR_SHMPAGES = 4,
  PCR_NUMPROC = 5,
  PCR_PHYSPAGES = 6,
  PCR_VMGUARPAGES = 7,
  PCR_OOMGUARPAGES = 8,
  PCR_NUMTCPSOCK = 9,
  PCR_NUMFLOCK = 10,
  PCR_NUMPTY = 11,
  PCR_NUMSIGINFO = 12,
  PCR_TCPSNDBUF = 13,
  PCR_TCPRCVBUF = 14,
  PCR_OTHERSOCKBUF = 15,
  PCR_DGRAMRCVBUF = 16,
  PCR_NUMOTHERSOCK = 17,
  PCR_DCACHESIZE = 18,
  PCR_NUMFILE = 19,
  PCR_NUMIPTENT = 20,
  PCR_SWAPPAGES = 21,
  PCR_QUOTAUGIDLIMIT = 22,
  PCR_LAST = PCR_QUOTAUGIDLIMIT
} PRL_CT_RESOURCE;
```

File

PrlEnums.h

PRL_CT_TEMPLATE_TYPE Enumeration

Container templates types

Syntax

```
typedef enum _PRL_CT_TEMPLATE_TYPE {
  PCT_TYPE_EZ_OS = 0,
  PCT_TYPE_EZ_APP = 1
} PRL_CT_TEMPLATE_TYPE;
```

File

PrlEnums.h

PRL_DATA_STATISTIC_SEGMENTS Enumeration

Data statistic segments

Syntax

```
typedef enum _PRL_DATA_STATISTIC_SEGMENTS {
  PDSS_UNKNOWN = 0,
  PDSS_VM_MISCELLANEOUS_SPACE = 1,
  PDSS_VM_RECLAIM_SPACE = 2,
  PDSS_VM_SNAPSHOTS_SPACE = 3,
  PDSS_VM_DISK_DATA_SPACE = 4,
  PDSS_VM_FULL_SPACE = 5
} PRL_DATA_STATISTIC_SEGMENTS;
```

File

PrlEnums.h

PRL_DELETE_SNAPSHOT_FLAGS Enumeration

- PrlVm_DeleteSnapshotEx flags PDSF_BACKUP - delete snapshot for backup

Syntax

```
typedef enum _PRL_DELETE_SNAPSHOT_FLAGS {
  PDSF_SKIP_SPACE_CHECK = 1<<(PACF_MAX+1),
  PDSF_BACKUP = 1<<(PACF_MAX+2),
  PDSF_BACKUP_MAP = 1<<(PACF_MAX+3)
} PRL_DELETE_SNAPSHOT_FLAGS;
```

File

PrlCommandsFlags.h

Members

PDSF_SKIP_SPACE_CHECK

skip space check

PDSF_BACKUP

internal flag

PDSF_BACKUP_MAP

internal flag

PRL_DEVICE_ACTION Enumeration

Device in use Action

Syntax

```
typedef enum _PRL_DEVICE_ACTION {
  PDA_DEVICE_IDLE = 0,
  PDA_DEVICE_READ = 1,
  PDA_DEVICE_WRITE = 2
```

```
} PRL_DEVICE_ACTION;
```

File

PrlEnums.h

PRL_DEVICE_TYPE Enumeration

Device type

Syntax

```
typedef enum _PRL_DEVICE_TYPE {
  PDE_GENERIC_DEVICE = 0,
  PDE_CLUSTERED_DEVICE = 1,
  PDE_STORAGE_DEVICE = 2,
  PDE_FLOPPY_DISK = 3,
  PDE_MASSSTORAGE_DEVICE = 4,
  PDE_OPTICAL_DISK = 5,
  PDE_HARD_DISK = 6,
  PDE_GENERIC_NETWORK_ADAPTER = 8,
  PDE_GENERIC_PORT = 9,
  PDE_SERIAL_PORT = 10,
  PDE_PARALLEL_PORT = 11,
  PDE_SOUND_DEVICE = 12,
  PDE_MIXER_DEVICE = 13,
  PDE_USB_DEVICE = 15,
  PDE_PRINTER = 16,
  PDE_GENERIC_PCI_DEVICE = 17,
  PDE_GENERIC_SCSI_DEVICE = 18,
  PDE_VIRTUAL_SNAPSHOT_DEVICE = 19,
  PDE_PCI_VIDEO_ADAPTER = 20,
  PDE_VIRTUAL_SHARED_FOLDERS_DEVICE = 21,
  PDE_ATTACHED_BACKUP_DISK = 22,
  PDE_MAX
} PRL_DEVICE_TYPE;
```

File

PrlEnums.h

PRL_DISK_PARAMETERS Structure

Virtuozzo disk parameters.

Syntax

```
typedef struct __PRL_DISK_PARAMETERS {
  PRL_UINT32 m_Heads;
  PRL_UINT32 m_Cylinders;
  PRL_UINT32 m_Sectors;
  PRL_UINT64 m_SizeInSectors;
  PRL_UINT32 m_BlockSize;
  PRL_UINT32 m_Parts;
  PRL_IMAGE_PARAMETERS_PTR m_Storages;
  PRL_GUID Uid;
  PRL_CHAR_PTR Name;
} PRL_DISK_PARAMETERS;
```

File

PrlDisk.h

Members

m_Storages

Following parameters are available only at SDK v3.

Remarks

Differs from internal due to lists classes

PRL_DISPLAY_BUFFER_FORMAT Enumeration

Format of display buffer

Syntax

```
typedef enum _PRL_DISPLAY_BUFFER_FORMAT {
  PDBF_NO_CONVERSION = 0,
  PDBF_INDEX8 = 1,
  PDBF_RGB15 = 2,
  PDBF_RGB16 = 3,
  PDBF_RGB24 = 4,
  PDBF_BGR32 = 5,
  PDBF_RGB32 = 6,
  PDBF_YUV422x8 = 10,
  PDBF_YUV422x6 = 11,
  PDBF_YUV422x4 = 12,
  PDBF_YUV420 = 16
} PRL_DISPLAY_BUFFER_FORMAT;
```

File

PrlEnums.h

PRL_DISPLAY_CODEC_COMPRESSION Enumeration

Obsolete enum.

Syntax

```
typedef enum _PRL_DISPLAY_CODEC_COMPRESSION {
  PDCC_HIGH_COMPRESSION = 1<<15,
  PDCC_MEDIUM_COMPRESSION = 1<<16,
  PDCC_LOW_COMPRESSION = 1<<17,
  PDCC_NO_COMPRESSION = 1<<18
} PRL_DISPLAY_CODEC_COMPRESSION;
```

File

PrlEnums.h

Remarks

Use display encodings at runtime for connection with new server

Notes

#PrlSrv_GetDisplayEncodingList Display codec compression

PRL_DISPLAY_CODEC_QUALITY Enumeration

Obsolete enum.

Syntax

```
typedef enum _PRL_DISPLAY_CODEC_QUALITY {
  PDCQ_HIGH_QUALITY = 1<<0,
  PDCQ_MEDIUM_QUALITY = 1<<1,
  PDCQ_LOW_QUALITY = 1<<2
} PRL_DISPLAY_CODEC_QUALITY;
```

File

PrlEnums.h

Remarks

Use display encodings at runtime for connection with new server

Notes

#PrlSrv_GetDisplayEncodingList Display codec quality.

PRL_DISPLAY_CODEC_TYPE Enumeration

Obsolete enum.

Syntax

```
typedef enum _PRL_DISPLAY_CODEC_TYPE {
  PDCT_HIGH_QUALITY_WITHOUT_COMPRESSION = PDCQ_HIGH_QUALITY|PDCC_NO_COMPRESSION,
  PDCT_HIGH_QUALITY_WITH_COMPRESSION = PDCQ_HIGH_QUALITY|PDCC_MEDIUM_COMPRESSION,
  PDCT_MEDIUM_QUALITY_WITHOUT_COMPRESSION = PDCQ_MEDIUM_QUALITY|PDCC_NO_COMPRESSION,
  PDCT_MEDIUM_QUALITY_WITH_COMPRESSION = PDCQ_MEDIUM_QUALITY|PDCC_MEDIUM_COMPRESSION,
  PDCT_LOW_QUALITY_WITHOUT_COMPRESSION = PDCQ_LOW_QUALITY|PDCC_NO_COMPRESSION,
  PDCT_LOW_QUALITY_WITH_COMPRESSION = PDCQ_LOW_QUALITY|PDCC_MEDIUM_COMPRESSION
} PRL_DISPLAY_CODEC_TYPE;
```

File

PrlEnums.h

Remarks

Use display encodings at runtime for connection with new server

Notes

#PrlSrv_GetDisplayEncodingList Display codec types

PRL_DISPLAY_CONFIG_MODE Enumeration

Display(s) configuration mode

Syntax

```
typedef enum _PRL_DISPLAY_CONFIG_MODE {
  PDCM_CLIENT_DISPLAY_INFO = (1 << 0),
  PDCM_HAS_HIDPI_DISPLAY = (1 << 1),
  PDCM_DO_NOT_RESTORE = (1 << 2),
  PDCM_RESTORE_DEFAULT = (1 << 3),
  PDCM_DISABLE_HIDPI_ADJUSTMENT = (1 << 4),
  PDCM_RETINA_MODE = (1 << 5),
  PDCM_ENABLE_HEADS = (1 << 6)
} PRL_DISPLAY_CONFIG_MODE;
```

File

PrlEnums.h

PRL_DISTRO_MEDIA_TYPE Enumeration

Syntax

```
typedef enum _PRL_DISTRO_MEDIA_TYPE {
  DMT_DISTRO = 0,
  DMT_LIVE_CD,
  DMT_MBR
} PRL_DISTRO_MEDIA_TYPE;
```

File

PrlEnums.h

Members

DMT_DISTRO

Ordinary distro

DMT_LIVE_CD

LiveCD

DMT_MBR

Media has Master Boot Record

PRL_DOCK_ICON_TYPE Enumeration

VM application icon type in Dock

Syntax

```
typedef enum _PRL_DOCK_ICON_TYPE {
  PIT_DOCK_ICON_SYSTEM = 0,
  PIT_DOCK_ICON_LIVE_SCREEN_SHOT = 1,
  PIT_DOCK_ICON_START_MENU = 2
} PRL_DOCK_ICON_TYPE;
```

File

PrlEnums.h

PRL_END_VM_BACKUP_FLAGS Enumeration

PrlVmBackup_Commit / PrlVmBackup_Rollback flags.

Syntax

```
typedef enum _PRL_END_VM_BACKUP_FLAGS {
  PEMBF_FAILURE_CLEANUP = 1<<(PACF_MAX+1)
} PRL_END_VM_BACKUP_FLAGS;
```

File

PrlCommandsFlags.h

Members

PEMBF_FAILURE_CLEANUP

Perform failure specific clean-up

PRL_EVENT_ISSUER_TYPE Enumeration

Event issuer type

Syntax

```
typedef enum _PRL_EVENT_ISSUER_TYPE {
  PIE_VIRTUAL_MACHINE = 0,
  PIE_DISPATCHER = 1,
  PIE_IO_SERVICE = 2,
  PIE_WEB_SERVICE = 3,
  PIE_UNKNOWN = 0xFF
} PRL_EVENT_ISSUER_TYPE;
```

File

PrlEnums.h

PRL_EVENT_TYPE Enumeration

Different event types are identified by these event codes.

Syntax

```
typedef enum _PRL_EVENT_TYPE {
  PET_VM_INF_UNINITIALIZED_EVENT_CODE = 100000,
  PET_DSP_EVT_VM_STATE_CHANGED = 100001,
  PET_DSP_EVT_VM_CONFIG_CHANGED = 100002,
  PET_DSP_EVT_VM_DEV_STATE_CHANGED = 100003,
  PET_DSP_EVT_VM_CREATED = 100004,
  PET_DSP_EVT_VM_ADDED = 100005,
  PET_DSP_EVT_VM_DELETED = 100006,
  PET_DSP_EVT_VM_STARTED = 100007,
  PET_DSP_EVT_VM_STOPPED = 100008,
  PET_DSP_EVT_VM_ABORTED = 100009,
  PET_DSP_EVT_VM_RESETED = 100010,
  PET_DSP_EVT_VM_SUSPENDING = 100011,
```

```
    PET_DSP_EVT_VM_SUSPENDED = 100012,
    PET_DSP_EVT_VM_RESUMING = 100013,
    PET_DSP_EVT_VM_RESUMED = 100014,
    PET_DSP_EVT_VM_START_IO_FLOW = 100015,
    PET_DSP_EVT_VM_SNAPSHOTED = 100016,
    PET_DSP_EVT_VM_STATESAVED = PET_DSP_EVT_VM_SNAPSHOTED,
    PET_DSP_EVT_VM_RESTORED = 100017,
    PET_DSP_EVT_VM_STATERESTORED = PET_DSP_EVT_VM_RESTORED,
    PET_DSP_EVT_VM_NET_RECONNECTED = 100018,
    PET_DSP_EVT_VM_PAUSED = 100019,
    PET_DSP_EVT_VM_CONTINUED = 100020,
    PET_DSP_EVT_VM_CONFIG_APPLIED = 100021,
    PET_DSP_EVT_VM_STATISTICS_UPDATED = 100022,
    PET_DSP_EVT_VM_PROBLEM_REPORT_CREATED = 100023,
    PET_DSP_EVT_VM_ABOUT_TO_RESET = 100024,
    PET_DSP_EVT_VM_SECURITY_CHANGED = 100025,
    PET_DSP_EVT_VM_UNREGISTERED = 100026,
    PET_DSP_EVT_VM_REMOTE_DEV_READY_TO_START = 100027,
    PET_VM_EVT_VM_CONFIG_CHANGED = 100028,
    PET_VM_EVT_REQUEST_FILE_OPEN = 100029,
    PET_DSP_EVT_VM_MIGRATE_STARTED = 100030,
    PET_DSP_EVT_VM_MIGRATE_CANCELLED = 100031,
    PET_DSP_EVT_VM_MIGRATE_FINISHED = 100032,
    PET_DSP_EVT_VM_MIGRATE_PROGRESS_CHANGED = 100033,
    PET_DSP_EVT_VM_SNAPSHOTING = 100034,
    PET_DSP_EVT_VM_RESTORING = 100035,
    PET_DSP_EVT_VM_SNAPSHOTS_TREE_CHANGED = 100036,
    PET_DSP_EVT_VM_DELETING_STATE = 100037,
    PET_DSP_EVT_VM_STATE_DELETED = 100038,
    PET_DSP_EVT_VM_ABOUT_TO_START_DEINIT = 100039,
    PET_DSP_EVT_VM_ABOUT_TO_START_INIT = 100040,
    PET_DSP_EVT_VM_MEMORY_SWAPPING_STARTED = 100041,
    PET_DSP_EVT_VM_MEMORY_SWAPPING_FINISHED = 100042,
    PET_VM_EVT_UNMOUNT = 100043,
    PET_VM_EVT_MOUNT = 100044,
    PET_DSP_EVT_REBOOT_HOST = 100045,
    PET_DSP_EVT_REBOOT_HOST_MANUAL_TO_INIT_VTD_DEVICES = 100046,
    PET_DSP_EVT_BACKUP_STARTED = 100047,
    PET_DSP_EVT_BACKUP_CANCELLED = 100048,
    PET_DSP_EVT_CREATE_BACKUP_FINISHED = 100049,
    PET_DSP_EVT_BACKUP_PROGRESS_CHANGED = 100050,
    PET_DSP_EVT_JOB_PROGRESS_CHANGED = 100051,
    PET_DSP_EVT_RESTORE_PROGRESS_CHANGED = 100052,
    PET_DSP_EVT_DISK_RESIZE_STARTED = 100053,
    PET_DSP_EVT_DISK_RESIZE_FINISHED = 100054,
    PET_DSP_EVT_DISK_RESIZE_PROGRESS_CHANGED = 100055,
    PET_DSP_EVT_VM_ADDITION_STATE_CHANGED = 100056,
    PET_DSP_EVT_RESTORE_BACKUP_FINISHED = 100057,
    PET_DSP_EVT_REMOVE_BACKUP_FINISHED = 100058,
    PET_DSP_EVT_VM_MIGRATE_CANCELLED_DISP = 100059,
    PET_DSP_EVT_VM_MIGRATE_FINISHED_DISP = 100060,
    PET_DSP_EVT_TIME_MACHINE_BACKUP_IS_OVER = 100061,
    PET_DSP_EVT_CONVERSION_DISKS_PROGRESS_CHANGED = 100062,
    PET_DSP_EVT_CONVERSION_DISKS_PROGRESS_FINISHED = 100063,
    PET_DSP_EVT_CONVERT_THIRD_PARTY_PROGRESS_CHANGED = 100064,
    PET_DSP_EVT_CONNECTED_TO_PROXY = 100065,
    PET_DSP_EVT_DISCONNECTED_FROM_PROXY = 100066,
    PET_DSP_EVT_CONNECTING_TO_PROXY = 100067,
    PET_DSP_EVT_RECONFIG_VM_PROGRESS_CHANGED = 100068,
    PET_DSP_EVT_VM_MIGRATE_WAIT_REMOUNT = 100069,
    PET_DSP_EVT_VM_CONFIG_APPLY_FINISHED = 100070,
    PET_DSP_EVT_VM_IO_CLIENT_STATISTICTS = 100071,
    PET_DSP_EVT_VM_SET_HOST_TIME = 100072,
    PET_DSP_EVT_VM_LOW_HOST_MEMORY = 100073,
    PET_DSP_EVT_VM_MAC_OS_VER = 100074,
    PET_DSP_EVT_VM_PAUSED_BY_HOST_SLEEP = 100075,
```

```
    PET_DSP_EVT_VM_CONTINUED_BY_HOST_WAKEUP = 100076,
    PET_DSP_EVT_VM_FROZEN = 100077,
    PET_DSP_EVT_VM_UNFROZEN = 100078,
    PET_DSP_EVT_CONNECTION_CLIENT_INFO_WAS_SET = 100079,
    PET_DSP_EVT_CONNECTION_WAS_CLOSED = 100080,
    PET_DSP_EVT_OP_ACCEPTED = 100100,
    PET_DSP_EVT_OP_REJECTED = 100101,
    PET_DSP_EVT_OP_RESULT = 100102,
    PET_DSP_EVT_DIR_STATE_CHANGED = 100200,
    PET_DSP_EVT_HOST_STATISTICS_UPDATED = 100201,
    PET_DSP_EVT_HW_CONFIG_CHANGED = 100202,
    PET_DSP_EVT_CLIENT_CONNECTED = 100300,
    PET_DSP_EVT_CLIENT_DISCONNECTED = 100301,
    PET_DSP_EVT_CLIENT_REJECTED = 100302,
    PET_DSP_EVT_FOUND_LOST_VM_CONFIG = 100400,
    PET_DSP_EVT_HAS_EVENT = 100401,
    PET_DSP_EVT_RECONNECT = 100402,
    PET_DSP_EVT_DISP_CONNECTION_CLOSED = 100403,
    PET_DSP_EVT_DISP_SHUTDOWN = 100404,
    PET_DSP_EVT_USER_PROFILE_CHANGED = 100500,
    PET_DSP_EVT_COMMON_PREFS_CHANGED = 100501,
    PET_DSP_EVT_VM_MESSAGE = 100502,
    PET_DSP_EVT_VM_QUESTION = 100503,
    PET_JOB_DELETE_VM_PROGRESS_CHANGED = 100504,
    PET_VM_INF_START_FILE_COPYING = 100505,
    PET_VM_INF_END_FILE_COPYING = 100506,
    PET_DSP_EVT_ERROR_MESSAGE = 100507,
    PET_DSP_EVT_WARNING_MESSAGE = 100508,
    PET_JOB_HDD_CREATE_PROGRESS_CHANGED = 100509,
    PET_JOB_FILE_COPY_PROGRESS_CHANGED = 100510,
    PET_JOB_SUSPEND_PROGRESS_CHANGED = 100511,
    PET_JOB_RESUME_PROGRESS_CHANGED = 100512,
    PET_DSP_EVT_ANSWER_TO_VM_WAS_DONE = 100513,
    PET_DSP_EVT_LICENSE_CHANGED = 100514,
    PET_JOB_CREATE_SNAPSHOT_PROGRESS_CHANGED = 100515,
    PET_JOB_SWITCH_TO_SNAPSHOT_PROGRESS_CHANGED = 100516,
    PET_JOB_DELETE_SNAPSHOT_PROGRESS_CHANGED = 100517,
    PET_JOB_COMMIT_UNFINISHED_DISK_OP_PROGRESS_CHANGED = 100518,
    PET_DSP_EVT_COMMIT_UNFINISHED_DISK_OP_STARTED = 100519,
    PET_DSP_EVT_COMMIT_UNFINISHED_DISK_OP_FINISHED = 100520,
    PET_DSP_EVT_VM_ENCRYPT_PROGRESS_CHANGED = 100521,
    PET_DSP_EVT_VM_DECRYPT_PROGRESS_CHANGED = 100522,
    PET_JOB_STAGE_PROGRESS_CHANGED = 100523,
    PET_VM_INF_START_BUNCH_COPYING = 100524,
    PET_VM_INF_END_BUNCH_COPYING = 100525,
    PET_DSP_EVT_SMC_USER_FORCE_DISCONNECTED = 100600,
    PET_DSP_EVT_SMC_CANCEL_USER_COMMAND = 100601,
    PET_DSP_EVT_SMC_DISPATCHER_SHUTDOWN = 100602,
    PET_IO_SCREEN_SIZE = 100700,
    PET_IO_SLIDING_MOUSE_FLAG = 100702,
    PET_IO_DEVICE_IS_IN_USE = 100703,
    PET_IO_KEYBOARD_LEDS = 100704,
    PET_IO_STATE = 100705,
    PET_IO_SCREEN_CAPTURED = 100706,
    PET_IO_BEFORE_SCREEN_SIZE = 100707,
    PET_IO_REMOTE_COMMAND = 100708,
    PET_IO_MOUSE_CURSOR_CHANGED = 100709,
    PET_IO_MOUSE_CURSOR_HID = 100710,
    PET_IO_MOUSE_CURSOR_SET = 100711,
    PET_IO_DYNRES_TOOL_STATUS = 100712,
    PET_IO_TOOLS_UTILITY_COMMAND = 100713,
    PET_IO_TOOLS_VM_SHUTDOWN = 100714,
    PET_IO_TOOLS_CLIPBOARD_DATA = 100715,
    PET_IO_SCREEN_SURFACE_DETACHED = 100716,
    PET_IO_TOOLS_SIA_DATA = 100717,
    PET_IO_TOOLS_VMCTG_COMMAND = 100718,
```

```
PET_IO_TOOLS_DRAGDROP_DATA = 100719,
PET_IO_TOOLS_GENERAL_COMMAND = 100720,
PET_IO_TOOLS_DESKTOP_UTILITY_STATE = 100721,
PET_IO_DISPLAY_SCREEN_SIZE = 100722,
PET_IO_DISPLAY_BEFORE_SCREEN_SIZE = 100723,
PET_IO_DISPLAY_SCREEN_CAPTURED = 100724,
PET_IO_SWITCH_VESA_MODE = 100725,
PET_IO_TOOLS_LANGUAGE_HOTKEY_CHANGED = 100726,
PET_IO_AVAILABLE_DISPLAYS = 100727,
PET_IO_SCREEN_SURFACE_ATTACHED = 100728,
PET_IO_UIEMU_ELEMENT_AT_POS = 100729,
PET_IO_AUDIO_OUTPUT_DATA = 100730,
PET_IO_AUDIO_OUTPUT_ENCODING_SET = 100731,
PET_IO_AUDIO_OUTPUT_STREAM_STARTED = 100732,
PET_IO_AUDIO_OUTPUT_STOP = 100734,
PET_IO_CVSRC_OPEN = 100735,
PET_IO_CVSRC_CLOSE = 100736,
PET_IO_SSO_GET_CREDENTIALS = 100737,
PET_IO_MOUSE_CURSOR_MOVED = 100738,
PET_IO_UIEMU_CARET_INFO = 100739,
PET_IO_DISPLAY_GAMMA_CHANGED = 100740,
PET_DSP_EVT_VM_TOOLS_STATE_CHANGED = 100800,
PET_DSP_EVT_VM_COMPACT_PROCESSING = 100801,
PET_DSP_EVT_VM_COMPACT_FINISHED = 100802,
PET_DSP_EVT_VM_COMPRESSOR_FINISHED = 100803,
PET_DSP_EVT_VM_COMPRESSOR_CONTINUED = 100804,
PET_DSP_EVT_VM_SOFTWARE_INSTALLED = 100805,
PET_DSP_EVT_VM_PIS_NOTIFICATION_ASK_TO_INSTALL = 100806,
PET_DSP_EVT_VM_VIRTUAL_DEVICES_STATE_CHANGED = 100807,
PET_DSP_EVT_PERFSTATS = 100900,
PET_DSP_EVT_VM_PERFSTATS = 100901,
PET_DSP_EVT_VM_UPGRADE_INIT = 101000,
PET_DSP_EVT_VM_UPGRADE_INIT_TIMEOUT = 101001,
PET_DSP_EVT_VM_UPGRADE_STAGE_1 = 101002,
PET_DSP_EVT_VM_UPGRADE_STAGE_2 = 101003,
PET_DSP_EVT_VM_UPGRADE_STAGE_3 = 101004,
PET_DSP_EVT_VM_UPGRADE_COMPLETED = 101005,
PET_DSP_EVT_VM_UPGRADE_UNKNOWN_ERROR = 101006,
PET_DSP_EVT_VM_UPGRADE_INIT_VMW = 101007,
PET_DSP_EVT_VM_UPGRADE_INIT_PHY = 101008,
PET_IO_READY_TO_ACCEPT_STDIN_PKGS = 101100,
PET_VM_READY_TO_BOOT = 101200,
PET_VM_READY_TO_PRINT = 101201,
PET_VM_DEFAULT_SOUND_RECONNECTED = 101202,
PET_VM_EVT_UNMOUNT_USB_MEDIA = 101203,
PET_VM_EVT_UNMOUNT_DVD_MEDIA = 101204,
PET_DSP_EVT_APPLIANCE_DOWNLOAD_PROGRESS_CHANGED = 101300,
PET_DSP_EVT_APPLIANCE_DOWNLOAD_FINISHED = 101301,
PET_DSP_EVT_APPLIANCE_ARCHIVE_UNPACK_STARTED = 101302,
PET_DSP_EVT_APPLIANCE_ARCHIVE_UNPACK_FINISHED = 101303,
PET_DSP_EVT_APPLIANCE_REGISTER_VM_STARTED = 101304,
PET_DSP_EVT_APPLIANCE_REGISTER_VM_FINISHED = 101305,
PET_DSP_EVT_CEP_WRITE_STRING = 101400,
PET_DSP_COHERENCE_FAKE_STUB_NOTIFICATION = 101401,
PET_DSP_EVT_VM_WAS_ENCRYPTED = 101500,
PET_DSP_EVT_VM_WAS_DECRYPTED = 101501,
PET_DSP_EVT_ENCRYPTED_VM_PASSWORD_CHANGED = 101502,
PET_DSP_EVT_ENCRYPTED_VM_NEED_AUTH_AGAIN = 101503,
PET_DSP_EVT_VM_WAS_PROTECTED = 101504,
PET_DSP_EVT_VM_WAS_UNPROTECTED = 101505,
PET_VM_EVT_TIS_BACKUP_WRITE = 101600,
PET_IO_EVT_DISCONNECT_WITH_REASON = 101700,
PET_VIDEO_RECEIVER_FORMAT = 101800,
PET_VIDEO_RECEIVER_FRAME = 101801,
PET_VIDEO_RECEIVER_CLOSED = 101802,
PET_VIDEO_CAPTURE_GEOMETRY = 101803,
```

```
    PET_VIDEO_RECEIVER_CONNECTION_STATE = 101804,
    PET_PTM_EVT_PROPERTY_CHANGED = 101900,
    PET_PTM_EVT_OBJECT_DELETED = 101901,
    PET_PTM_EVT_OBJECT_CREATED = 101902,
    PET_PTM_EVT_SUBSCRIBED = 101903,
    PET_PTM_EVT_RPC_COMPLETED = 101904
} PRL_EVENT_TYPE;
```

File

PrlEventsValues.h

Members

PET_VM_INF_UNINITIALIZED_EVENT_CODE
BASE VALUE SET TO 100000

PET_DSP_EVT_OP_ACCEPTED
ALIGNED VALUE TO 100100

PET_DSP_EVT_DIR_STATE_CHANGED
ALIGNED VALUE TO 100200

PET_DSP_EVT_CLIENT_CONNECTED
ALIGNED VALUE TO 100300

PET_DSP_EVT_FOUND_LOST_VM_CONFIG
ALIGNED VALUE TO 100400

PET_DSP_EVT_USER_PROFILE_CHANGED
ALIGNED VALUE TO 100500

PET_DSP_EVT_SMC_USER_FORCE_DISCONNECTED
ALIGNED VALUE TO 100600

PET_IO_SCREEN_SIZE
ALIGNED VALUE TO 100700

PET_DSP_EVT_VM_TOOLS_STATE_CHANGED
ALIGNED VALUE TO 100800

PET_DSP_EVT_PERFSTATS
ALIGNED VALUE TO 100900

PET_DSP_EVT_VM_UPGRADE_INIT
ALIGNED VALUE TO 101000

PET_IO_READY_TO_ACCEPT_STDIN_PKGS
ALIGNED VALUE TO 101100

PET_VM_READY_TO_BOOT
ALIGNED VALUE TO 101200

PET_DSP_EVT_APPLIANCE_DOWNLOAD_PROGRESS_CHANGED
ALIGNED VALUE TO 101300

PET_DSP_EVT_CEP_WRITE_STRING
ALIGNED VALUE TO 101400

PET_DSP_EVT_VM_WAS_ENCRYPTED
ALIGNED VALUE TO 101500

PET_VM_EVT_TIS_BACKUP_WRITE
ALIGNED VALUE TO 101600

PET_IO_EVT_DISCONNECT_WITH_REASON

ALIGNED VALUE TO 101700

PET_VIDEO_RECEIVER_FORMAT
ALIGNED VALUE TO 101800

PET_PTM_EVT_PROPERTY_CHANGED
ALIGNED VALUE TO 101900

Remarks

Client can query event handle with the PrlEvent_GetType (p. 86) call to get this code. @see PrlEvent_GetType (p. 86)

PRL_EXTERNAL_VM_TYPE Enumeration

External VM type for supporting native VM

Syntax

```
typedef enum _PRL_EXTERNAL_VM_TYPE {
  PEVT_VM_UNKNOWN = 0,
  PEVT_NATIVE_VM = PEVT_VM_UNKNOWN,
  PEVT_VMWARE = 1
} PRL_EXTERNAL_VM_TYPE;
```

File

PrlEnums.h

PRL_FEATURES_MATRIX Enumeration

Features matrix list

Syntax

```
typedef enum _PRL_FEATURES_MATRIX {
  PFSM_UNKNOWN_FEATURE = 0,
  PFSM_SATA_HOTPLUG_SUPPORT = 1,
  PFSM_DEFAULT_SATA_ALLOWED = PFSM_SATA_HOTPLUG_SUPPORT,
  PFSM_VM_CONFIG_MERGE_SUPPORT = 2,
  PFSM_PSBM5 = 3,
  PFSM_AUTOSTART_VM_AS_OWNER = 4,
  PFSM_DEFAULT_PLAINDISK_ALLOWED = 5,
  PFSM_NO_SHARED_NETWORKING = 6,
  PFSM_DISK_IO_LIMITS = 7,
  PFSM_FINE_CPU_LIMITS = 8,
  PFSM_ROUTED_NETWORKING = 9,
  PFSM_RAM_HOTPLUG_SUPPORT = 10,
  PFSM_CPU_HOTPLUG_SUPPORT = 11,
  PFSM_IPV6_SUPPORT = 12,
  PFSM_USB_PRINTER_SUPPORT = 13,
  PFSM_NIC_CHANGE_ALLOWED = 14,
  PFSM_DESKTOP_CONTROL_SUPPORT = 15
} PRL_FEATURES_MATRIX;
```

File

PrlEnums.h

PRL_FILE_SYSTEM_ELEMENT_TYPE Enumeration

File system element type

Syntax

```
typedef enum _PRL_FILE_SYSTEM_ELEMENT_TYPE {
  PSE_DRIVE = 0,
  PSE_DIRECTORY = 1,
  PSE_FILE = 2
} PRL_FILE_SYSTEM_ELEMENT_TYPE;
```

File

PrlEnums.h

PRL_FILE_SYSTEM_FS_TYPE Enumeration

Syntax

```
typedef enum _PRL_FILE_SYSTEM_FS_TYPE {
  PRL_FS_INVALID = 0,
  PRL_FS_FAT = 1,
  PRL_FS_FAT32 = 2,
  PRL_FS_NTFS = 3,
  PRL_FS_HFS = 4,
  PRL_FS_ADFS = 5,
  PRL_FS_AFFS = 6,
  PRL_FS_AFS = 7,
  PRL_FS_AUTOFS = 8,
  PRL_FS_CODA = 9,
  PRL_FS_EFS = 10,
  PRL_FS_EXTFS = 11,
  PRL_FS_HPFS = 12,
  PRL_FS_ISOFS = 13,
  PRL_FS_JFFS2 = 14,
  PRL_FS_NFS = 15,
  PRL_FS_QNX4 = 16,
  PRL_FS_REISERFS = 17,
  PRL_FS_SMBFS = 18,
  PRL_FS_GFS = 19,
  PRL_FS_FUSE = 20,
  PRL_FS_UNSPECIFIED = 0xFF
} PRL_FILE_SYSTEM_FS_TYPE;
```

File

PrlEnums.h

PRL_FILE_SYSTEM_TYPE Enumeration

File system general type

Syntax

```
typedef enum _PRL_FILE_SYSTEM_TYPE {
  PFS_WINDOWS_LIKE_FS = 0,
  PFS_UNIX_LIKE_FS = 1
} PRL_FILE_SYSTEM_TYPE;
```

File

PrlEnums.h

PRL_FIREWALL_DIRECTION Enumeration

Firewall direction

Syntax

```
typedef enum _PRL_FIREWALL_DIRECTION {
  PFD_INCOMING = 0,
  PFD_OUTGOING = 1
} PRL_FIREWALL_DIRECTION;
```

File

PrlEnums.h

PRL_FIREWALL_POLICY Enumeration

Firewall policy

Syntax

```
typedef enum _PRL_FIREWALL_POLICY {
  PFP_ACCEPT = 0,
  PFP_DENY = 1
} PRL_FIREWALL_POLICY;
```

File

PrlEnums.h

PRL_GENERIC_DEVICE_STATE Enumeration

Generic device state PGS_CONNECTED_TO_HOST : device is used by host PGS_CONNECTED_TO_VM : device is used by VM PGS_RESERVED : reserved value PGS_NON_CONTROLLED_USB : USB device is not controlled PGS_CONNECTING_TO_VM : device belong to VM, but prl_usb_dev still not started

Syntax

```
typedef enum _PRL_GENERIC_DEVICE_STATE {
  PGS_CONNECTED_TO_HOST = 0,
  PGS_CONNECTED_TO_VM = 1,
  PGS_RESERVED = 2,
  PGS_NON_CONTROLLED_USB = PGS_RESERVED,
  PGS_CONNECTING_TO_VM = 3
```

```
} PRL_GENERIC_DEVICE_STATE;
```

File

PrlEnums.h

PRL_GENERIC_PCI_DEVICE_CLASS Enumeration

Generic PCI device class: PGD_PCI_NETWORK : generic PCI network device PGD_PCI_DISPLAY : generic PCI display device PGD_PCI_SOUND : generic PCI sound device PGD_PCI_OTHER : another generic PCI device

Syntax

```
typedef enum _PRL_GENERIC_PCI_DEVICE_CLASS {
  PGD_PCI_NETWORK = 0,
  PGD_PCI_DISPLAY = 1,
  PGD_PCI_SOUND = 2,
  PGD_PCI_OTHER = 3
} PRL_GENERIC_PCI_DEVICE_CLASS;
```

File

PrlEnums.h

PRL_GET_SNAPSHOT_TREE_FLAGS Enumeration

PrlVm_GetSnapshotsTreeEx (p. 468) flags

Syntax

```
typedef enum _PRL_GET_SNAPSHOT_TREE_FLAGS {
  PGST_FULL_SIZE_SCREENSHOTS = 1<<(PACF_MAX+1),
  PGST_WITHOUT_SCREENSHOTS = 1<<(PACF_MAX+2)
} PRL_GET_SNAPSHOT_TREE_FLAGS;
```

File

PrlCommandsFlags.h

Members

PGST_FULL_SIZE_SCREENSHOTS
return screenshots in full size

PGST_WITHOUT_SCREENSHOTS
fast way to request snapshots tree

PRL_GET_VM_CONFIG_FLAGS Enumeration

Find VM operations:

Syntax

```
typedef enum _PRL_GET_VM_CONFIG_FLAGS {
  PGVC_SEARCH_BY_UUID = 1<<(PACF_MAX+1),
  PGVC_SEARCH_BY_NAME = 1<<(PACF_MAX+2),
```

```
  PGVC_FILL_AUTOGENERATED = 1<<(PACF_MAX+3)
} PRL_GET_VM_CONFIG_FLAGS;
```

File

PrlCommandsFlags.h

PRL_GET_VM_LIST_FLAGS Enumeration

List operations: Uses for PrlSrv_GetVmListEx (p. 266) operation

Syntax

```
typedef enum _PRL_GET_VM_LIST_FLAGS {
  PGVLF_GET_ONLY_VM = PVTF_VM,
  PGVLF_GET_ONLY_CT = PVTF_CT,
  PGVLF_GET_ONLY_IDENTITY_INFO = 1<<(PACF_MAX+3),
  PGVLF_GET_STATE_INFO = 1<<(PACF_MAX+4),
  PGVLF_FILL_AUTOGENERATED = 1<<(PACF_MAX+5),
  PGVLF_GET_ONLY_VM_TEMPLATES = 1<<(PACF_MAX+6)
} PRL_GET_VM_LIST_FLAGS;
```

File

PrlCommandsFlags.h

PRL_GRACEFUL_SHUTDOWN_CAPABILITIES Enumeration

Syntax

```
typedef enum _PRL_GRACEFUL_SHUTDOWN_CAPABILITIES {
  PGST_CAPABILITY_SHUTDOWN = 0x1,
  PGST_CAPABILITY_REBOOT = 0x2,
  PGST_CAPABILITY_SUSPEND = 0x4,
  PGST_CAPABILITY_HIBERNATE = 0x8
} PRL_GRACEFUL_SHUTDOWN_CAPABILITIES;
```

File

PrlEnums.h

PRL_GRACEFUL_SHUTDOWN_KIND Enumeration

Graceful Shutdown types

Syntax

```
typedef enum _PRL_GRACEFUL_SHUTDOWN_KIND {
  PST_VM_SHUTDOWN = 0,
  PST_VM_REBOOT = 1,
  PST_VM_HIBERNATE = 2,
  PST_VM_SUSPEND = 3
} PRL_GRACEFUL_SHUTDOWN_KIND;
```

File

PrlEnums.h

PRL_GUEST_DISTRO_VERSION_ENUM Enumeration

Windows installation distributive type.

Syntax

```
typedef enum _PRL_GUEST_DISTRO_VERSION_ENUM {
  PGD_UNKNOWN = 0,
  PGD_WINDOWS_VISTA = 1,
  PGD_WINDOWS_XP = 2,
  PGD_WINDOWS_XP_32 = PGD_WINDOWS_XP,
  PGD_WINDOWS_XP_64 = 3,
  PGD_WINDOWS_VISTA_32 = PGD_WINDOWS_VISTA,
  PGD_WINDOWS_VISTA_64 = 4
} PRL_GUEST_DISTRO_VERSION_ENUM;
```

File

PrlEnums.h

PRL_GUEST_OS_SUPPORT_TYPE Enumeration

Specifies the type of guest OS's support

Syntax

```
typedef enum _PRL_GUEST_OS_SUPPORT_TYPE {
  PGS_OS_SUPPORTED = 0,
  PGS_OS_EXPERIMENTAL = 1
} PRL_GUEST_OS_SUPPORT_TYPE;
```

File

PrlEnums.h

PRL_HANDLE_TYPE Enumeration

The _PRL_HANDLE_TYPE (p. 996) enumeration defines handle types.

Syntax

```
typedef enum _PRL_HANDLE_TYPE {
  PHT_ERROR = 0,
  PHT_SERVER = 0x10000001,
  PHT_SERVER_CONFIG = 0x10000002,
  PHT_REMOTE_FILESYSTEM_INFO = 0x10000003,
  PHT_REMOTE_FILESYSTEM_ENTRY = 0x10000004,
  PHT_VIRTUAL_MACHINE = 0x10000005,
  PHT_VIRTUAL_DEV_DISPLAY = 0x10000006,
  PHT_VIRTUAL_DEV_KEYBOARD = 0x10000007,
  PHT_VIRTUAL_DEV_MOUSE = 0x10000008,
  PHT_VIRTUAL_DEV_FLOPPY = 0x10000009,
  PHT_VIRTUAL_DEV_HARD_DISK = 0x1000000A,
  PHT_VIRTUAL_DEV_NET_ADAPTER = 0x1000000B,
  PHT_VIRTUAL_DEV_PARALLEL_PORT = 0x1000000C,
  PHT_VIRTUAL_DEV_SERIAL_PORT = 0x1000000D,
```

```
    PHT_VIRTUAL_DEV_OPTICAL_DISK = 0x1000000E,
    PHT_VIRTUAL_DEV_USB_DEVICE = 0x1000000F,
    PHT_VIRTUAL_DEV_SOUND = 0x10000010,
    PHT_JOB = 0x10000011,
    PHT_EVENT = 0x10000012,
    PHT_RESULT = 0x10000013,
    PHT_VM_INFO = 0x10000014,
    PHT_USER_PROFILE = 0x10000015,
    PHT_DISP_CONFIG = 0x10000016,
    PHT_SHARE = 0x10000017,
    PHT_SCREEN_RESOLUTION = 0x10000018,
    PHT_DISP_NET_ADAPTER = 0x10000019,
    PHT_HW_GENERIC_DEVICE = 0x1000001A,
    PHT_HW_HARD_DISK = 0x1000001B,
    PHT_HW_HARD_DISK_PARTITION = 0x1000001C,
    PHT_HW_NET_ADAPTER = 0x1000001D,
    PHT_EVENT_PARAMETER = 0x1000001E,
    PHT_REMOTEDEV_CMD = 0x1000001F,
    PHT_SYSTEM_STATISTICS = 0x10000020,
    PHT_SYSTEM_STATISTICS_CPU = 0x10000021,
    PHT_SYSTEM_STATISTICS_IFACE = 0x10000022,
    PHT_SYSTEM_STATISTICS_USER_SESSION = 0x10000023,
    PHT_SYSTEM_STATISTICS_DISK = 0x10000024,
    PHT_SYSTEM_STATISTICS_DISK_PARTITION = 0x10000025,
    PHT_SYSTEM_STATISTICS_PROCESS = 0x10000026,
    PHT_LICENSE = 0x10000027,
    PHT_SERVER_INFO = 0x10000028,
    PHT_BOOT_DEVICE = 0x10000029,
    PHT_STRINGS_LIST = 0x1000002A,
    PHT_LOGIN_RESPONSE = 0x1000002B,
    PHT_RUNNING_TASK = 0x1000002C,
    PHT_NET_SERVICE_STATUS = 0x1000002D,
    PHT_HANDLES_LIST = 0x1000002E,
    PHT_VIRTUAL_DISK = 0x1000002F,
    PHT_ACCESS_RIGHTS = 0x10000030,
    PHT_VM_TOOLS_INFO = 0x10000031,
    PHT_FOUND_VM_INFO = 0x10000032,
    PHT_TIS_RECORD = 0x10000033,
    PHT_USER_INFO = 0x10000034,
    PHT_VM_CONFIGURATION = PHT_VIRTUAL_MACHINE,
    PHT_TIS_EMITTER = 0x10000036,
    PHT_TOOL = 0x10000037,
    PHT_VIRTUAL_DEV_HD_PARTITION = 0x10000038,
    PHT_VIRTUAL_DEV_GENERIC_PCI = 0x10000039,
    PHT_VIRTUAL_DEV_GENERIC_SCSI = 0x1000003A,
    PHT_VM_GUEST_SESSION = 0x1000003B,
    PHT_VIRTUAL_NETWORK = 0x1000003C,
    PHT_PORT_FORWARDING = 0x1000003D,
    PHT_HW_GENERIC_PCI_DEVICE = 0x1000003E,
    PHT_OPAQUE_TYPE_LIST = 0x1000003F,
    PHT_GUEST_OSES_MATRIX = 0x10000040,
    PHT_PROBLEM_REPORT = 0x10000041,
    PHT_APPLIANCE_CONFIG = 0x10000042,
    PHT_OFFLINE_SERVICE = 0x10000043,
    PHT_VM_VIRTUAL_DEVICES_INFO = 0x10000044,
    PHT_NETWORK_CLASS = 0x10000045,
    PHT_NETWORK_SHAPING = 0x10000046,
    PHT_NETWORK_RATE = 0x10000047,
    PHT_ISCSI_LUN = 0x10000048,
    PHT_CT_TEMPLATE = 0x10000049,
    PHT_USB_IDENTITY = 0x10000051,
    PHT_FIREWALL_RULE = 0x10000052,
    PHT_IPPRIV_NET = 0x10000053,
    PHT_PLUGIN_INFO = 0x10000054,
    PHT_CVSRC = 0x10000055,
    PHT_DESKTOP_CONTROL = 0x10000056,
```

```
    PHT_BACKUP_RESULT = 0x10000057,
    PHT_NETWORK_SHAPING_CONFIG = 0x10000058,
    PHT_NETWORK_SHAPING_BANDWIDTH = 0x10000059,
    PHT_SYSTEM_STATISTICS_VM_DATA = 0x1000005A,
    PHT_VIDEO_RECEIVER = 0x1000005B,
    PHT_SESSION_INFO = 0x1000005C,
    PHT_CPU_FEATURES = 0x1000005D,
    PHT_CPU_POOL = 0x1000005E,
    PHT_VIRTUAL_DISK_MAP = 0x1000005F,
    PHT_VIRTUAL_DISK_OPEN_POLICY = 0x10000060,
    PHT_VIRTUAL_DISK_ENCRYPTION = 0x10000061,
    PHT_VCMMD_CONFIG = 0x10000062,
    PHT_LAST = PHT_VCMMD_CONFIG
} PRL_HANDLE_TYPE;
```

File

PrlEnums.h

Members

PHT_LAST

should be set to last

Remarks

Each handle type and the functions that operate on it is described in detail in the "Functions by Handle" chapter of the Virtuozzo C API Reference guide.

PRL_HARD_DISK_IFACE_MASK Enumeration

Hard disk interfce mask enum

Syntax

```
typedef enum _PRL_HARD_DISK_IFACE_MASK {
  PIM_IDE_0_0 = (0x1L<<0),
  PIM_IDE_0_1 = (0x1L<<1),
  PIM_IDE_1_0 = (0x1L<<2),
  PIM_IDE_1_1 = (0x1L<<3),
  PIM_SCSI_0_0 = (0x1L<<4),
  PIM_SCSI_1_0 = (0x1L<<5),
  PIM_SCSI_2_0 = (0x1L<<6),
  PIM_SCSI_3_0 = (0x1L<<7),
  PIM_SCSI_4_0 = (0x1L<<8),
  PIM_SCSI_5_0 = (0x1L<<9),
  PIM_SCSI_6_0 = (0x1L<<10),
  PIM_SCSI_7_0 = (0x1L<<11),
  PIM_SCSI_8_0 = (0x1L<<12),
  PIM_SCSI_9_0 = (0x1L<<13),
  PIM_SCSI_10_0 = (0x1L<<14),
  PIM_SCSI_11_0 = (0x1L<<15),
  PIM_SCSI_12_0 = (0x1L<<16),
  PIM_SCSI_13_0 = (0x1L<<17),
  PIM_SCSI_14_0 = (0x1L<<18),
  PIM_SCSI_15_0 = (0x1L<<19),
  PIM_SATA_0_0 = (0x1L<<20),
  PIM_SATA_0_1 = (0x1L<<21),
  PIM_SATA_0_2 = (0x1L<<22),
  PIM_SATA_0_3 = (0x1L<<23),
  PIM_SATA_0_4 = (0x1L<<24),
  PIM_SATA_0_5 = (0x1L<<25),
  PIM_ALL_DISKS = ((0x1L<<26) - 1),
```

```
  PIM_COMPACT_DELAYED = (0x1L<<31),
  PIM_IDE_MASK_OFFSET = PIM_IDE_0_0,
  PIM_SCSI_MASK_OFFSET = PIM_SCSI_0_0,
  PIM_SATA_MASK_OFFSET = PIM_SATA_0_0
} PRL_HARD_DISK_IFACE_MASK;
```

File

PrlEnums.h

PRL_HARD_DISK_INTERNAL_FORMAT Enumeration

Hard disk image type

Syntax

```
typedef enum _PRL_HARD_DISK_INTERNAL_FORMAT {
  PHD_PLAIN_HARD_DISK = 0,
  PHD_EXPANDING_HARD_DISK = 1
} PRL_HARD_DISK_INTERNAL_FORMAT;
```

File

PrlEnums.h

PRL_HOST_OS_TYPE Enumeration

Host OS type

Syntax

```
typedef enum _PRL_HOST_OS_TYPE {
  PHO_MAC = 0,
  PHO_LIN = 1,
  PHO_WIN = 2,
  PHO_UNKNOWN = 0xFF
} PRL_HOST_OS_TYPE;
```

File

PrlEnums.h

PRL_HW_INFO_NET_ADAPTER_TYPE Enumeration

Network adapter types in hardware info list

Syntax

```
typedef enum _PRL_HW_INFO_NET_ADAPTER_TYPE {
  PHI_REAL_NET_ADAPTER = 0,
  PHI_VIRTUAL_NET_ADAPTER = 1,
  PHY_WIFI_REAL_NET_ADAPTER = 2
} PRL_HW_INFO_NET_ADAPTER_TYPE;
```

File

PrlEnums.h

PRL_IMAGE_FORMAT Enumeration

Syntax

```
typedef enum _PRL_IMAGE_FORMAT {
  PIF_RAW = 0x50000008,
  PIF_BMP = 0x50000009,
  PIF_JPG = 0x5000000A,
  PIF_PNG = 0x5000000B
} PRL_IMAGE_FORMAT;
```

File

PrlEnums.h

PRL_IMAGE_PARAMETERS Structure

Virtuozzo storage parameters

Syntax

```
typedef struct __PRL_IMAGE_PARAMETERS {
  PRL_IMAGE_TYPE Type;
  PRL_UINT64 uStart;
  PRL_UINT64 uSize;
  PRL_CHAR_PTR pFileName;
} PRL_IMAGE_PARAMETERS;
```

File

PrlDisk.h

PRL_INSTALL_APPLIANCE_FLAGS Enumeration

Install appliance options

- PIAF_CANCEL : cancel appliance installation
- PIAF_FORCE : do not cancel installation and begin a new one
- if the previous installation became corrupted outside.
- PIAF_STOP : temporary stop appliance downloading

Syntax

```
typedef enum _PRL_INSTALL_APPLIANCE_FLAGS {
  PIAF_CANCEL = 1<<(PACF_MAX+1),
  PIAF_FORCE = 1<<(PACF_MAX+2),
  PIAF_STOP = 1<<(PACF_MAX+3)
} PRL_INSTALL_APPLIANCE_FLAGS;
```

File

PrlCommandsFlags.h

PRL_IOPRIO Enumeration

Syntax

```
typedef enum _PRL_IOPRIO {
  PRL_IOPRIO_MIN = 0,
  PRL_IOPRIO_DEFAULT = 4,
  PRL_IOPRIO_MAX = 7
} PRL_IOPRIO;
```

File

PrlEnums.h

PRL_ISCSI_STORAGE_CMD_FLAGS Enumeration

iSCSI storage management flags

Syntax

```
typedef enum _PRL_ISCSI_STORAGE_CMD_FLAGS {
  PISCSI_STORAGE_CREATE = 1<<(PACF_MAX+1),
  PISCSI_STORAGE_REMOVE = 1<<(PACF_MAX+2),
  PISCSI_STORAGE_MOUNT_RDONLY = 1<<(PACF_MAX+3),
  PISCSI_STORAGE_REMOUNT = 1<<(PACF_MAX+4),
  PISCSI_STORAGE_MOUNT = 1<<(PACF_MAX+5)
} PRL_ISCSI_STORAGE_CMD_FLAGS;
```

File

PrlCommandsFlags.h

Members

PISCSI_STORAGE_CREATE
To create storage on register

PISCSI_STORAGE_REMOVE
To remove storage on unregister

PISCSI_STORAGE_MOUNT_RDONLY
To mount storage by read-only

PISCSI_STORAGE_REMOUNT
To remount storage

PISCSI_STORAGE_MOUNT
To mount storage on register

PRL_ISCSI_STORAGE_FSTYPE_FLAGS Enumeration

iSCSI storage filesystem type

Syntax

```
typedef enum _PRL_ISCSI_STORAGE_FSTYPE_FLAGS {
  PISCSI_STORAGE_EXT3 = 1<<(PACF_MAX+1),
  PISCSI_STORAGE_EXT4 = 1<<(PACF_MAX+2),
  PISCSI_STORAGE_NTFS = 1<<(PACF_MAX+3)
} PRL_ISCSI_STORAGE_FSTYPE_FLAGS;
```

File

PrlCommandsFlags.h

PRL_JOB_OPERATION_CODE Enumeration

Job operation codes

Syntax

```
typedef enum _PRL_JOB_OPERATION_CODE {
  PJOC_UNKNOWN = 0,
  PJOC_JOB_CANCEL = 1,
  PJOC_SRV_LOOKUP_PARALLELS_SERVERS = 2,
  PJOC_SRV_LOGIN = 3,
  PJOC_SRV_LOGIN_LOCAL = 4,
  PJOC_SRV_LOGOFF = 5,
  PJOC_SRV_GET_SRV_CONFIG = 6,
  PJOC_SRV_GET_COMMON_PREFS = 7,
  PJOC_SRV_COMMON_PREFS_BEGIN_EDIT = 8,
  PJOC_SRV_COMMON_PREFS_COMMIT = 9,
  PJOC_SRV_GET_USER_PROFILE = 10,
  PJOC_SRV_GET_STATISTICS = 11,
  PJOC_SRV_USER_PROFILE_BEGIN_EDIT = 12,
  PJOC_SRV_USER_PROFILE_COMMIT = 13,
  PJOC_SRV_REGISTER_VM = 14,
  PJOC_SRV_REGISTER_VM_EX = PJOC_SRV_REGISTER_VM,
  PJOC_SRV_GET_VM_LIST = 15,
  PJOC_SRV_SUBSCRIBE_TO_HOST_STATISTICS = 16,
  PJOC_SRV_UNSUBSCRIBE_FROM_HOST_STATISTICS = 17,
  PJOC_SRV_SHUTDOWN = 18,
  PJOC_SRV_FS_GET_DISK_LIST = 19,
  PJOC_SRV_FS_GET_DIR_ENTRIES = 20,
  PJOC_SRV_FS_CREATE_DIR = 21,
  PJOC_SRV_FS_REMOVE_ENTRY = 22,
  PJOC_SRV_FS_CAN_CREATE_FILE = 23,
  PJOC_SRV_FS_RENAME_ENTRY = 24,
  PJOC_SRV_FS_GENERATE_ENTRY_NAME = 25,
  PJOC_SRV_UPDATE_LICENSE = 26,
  PJOC_SRV_GET_LICENSE_INFO = 27,
  PJOC_SRV_SEND_ANSWER = 28,
  PJOC_SRV_START_SEARCH_VMS = 29,
  PJOC_SRV_NET_SERVICE_START = 30,
  PJOC_SRV_NET_SERVICE_STOP = 31,
  PJOC_SRV_NET_SERVICE_RESTART = 32,
  PJOC_SRV_NET_SERVICE_RESTORE_DEFAULTS = 33,
  PJOC_SRV_GET_NET_SERVICE_STATUS = 34,
  PJOC_SRV_ADD_NET_ADAPTER = 35,
  PJOC_SRV_DELETE_NET_ADAPTER = 36,
  PJOC_SRV_UPDATE_NET_ADAPTER = 37,
  PJOC_SRV_GET_PROBLEM_REPORT = 38,
  PJOC_SRV_ATTACH_TO_LOST_TASK = 39,
  PJOC_SRV_GET_USER_INFO_LIST = 40,
  PJOC_SRV_GET_USER_INFO = 41,
  PJOC_VM_START = 42,
```

```
    PJOC_VM_STOP = 43,
    PJOC_VM_PAUSE = 44,
    PJOC_VM_RESET = 45,
    PJOC_VM_SUSPEND = 46,
    PJOC_VM_RESUME = PJOC_VM_START,
    PJOC_VM_DROP_SUSPENDED_STATE = 49,
    PJOC_VM_CLONE = 50,
    PJOC_VM_DELETE = 51,
    PJOC_VM_GET_PROBLEM_REPORT = 52,
    PJOC_VM_GET_STATE = 53,
    PJOC_VM_GENERATE_VM_DEV_FILENAME = 54,
    PJOC_VM_GET_TOOLS_STATE = 55,
    PJOC_VM_REFRESH_CONFIG = 56,
    PJOC_VM_GET_STATISTICS = 57,
    PJOC_VM_SUBSCRIBE_TO_GUEST_STATISTICS = 58,
    PJOC_VM_UNSUBSCRIBE_FROM_GUEST_STATISTICS = 59,
    PJOC_VM_REG = 60,
    PJOC_VM_UNREG = 61,
    PJOC_VM_BEGIN_EDIT = 62,
    PJOC_VM_COMMIT = 63,
    PJOC_VM_CREATE_UNATTENDED_FLOPPY = 64,
    PJOC_VM_INITIATE_DEV_STATE_NOTIFICATIONS = 65,
    PJOC_VM_UPDATE_SECURITY = 66,
    PJOC_VM_VALIDATE_CONFIG = 67,
    PJOC_VM_DEV_CONNECT = 68,
    PJOC_VM_DEV_DISCONNECT = 69,
    PJOC_VM_DEV_CREATE_IMAGE = 70,
    PJOC_VM_INSTALL_TOOLS = 71,
    PJOC_VM_SEND_SHUTDOWN_COMMAND = 72,
    PJOC_VM_SEND_CLIPBOARD_REQUEST = 73,
    PJOC_VM_CONNECT_TO_VM = 74,
    PJOC_VM_DEV_DISPLAY_CAPTURE_SCREEN = 75,
    PJOC_VM_MIGRATE = 76,
    PJOC_VM_SEND_DRAGDROP_COMMAND = 77,
    PJOC_VM_GUEST_RUN_PROGRAM = 78,
    PJOC_VM_GUEST_LOGOUT = 79,
    PJOC_SRV_SUBSCRIBE_PERFSTATS = 80,
    PJOC_SRV_UNSUBSCRIBE_PERFSTATS = 81,
    PJOC_SRV_GET_PERFSTATS = 82,
    PJOC_VM_SUBSCRIBE_PERFSTATS = 83,
    PJOC_VM_UNSUBSCRIBE_PERFSTATS = 84,
    PJOC_VM_GET_PERFSTATS = 85,
    PJOC_VM_GET_SUSPENDED_SCREEN = 86,
    PJOC_VM_MIGRATE_CANCEL = 87,
    PJOC_VM_RUN_COMPRESSOR = 88,
    PJOC_VM_CANCEL_COMPRESSOR = 89,
    PJOC_VM_CREATE_SNAPSHOT = 90,
    PJOC_VM_SWITCH_TO_SNAPSHOT = 91,
    PJOC_VM_DELETE_SNAPSHOT = 92,
    PJOC_VM_GET_SNAPSHOTS_TREE = 93,
    PJOC_VM_START_EX = 94,
    PJOC_VM_LOGIN_IN_GUEST = 95,
    PJOC_VM_UPDATE_SNAPSHOT_DATA = 96,
    PJOC_VM_RESTART = 97,
    PJOC_VM_INSTALL_UTILITY = 98,
    PJOC_VM_UPDATE_TOOLS_SECTION = 99,
    PJOC_SRV_GET_ALL_HOST_USERS = 100,
    PJOC_VM_RESTORE = 101,
    PJOC_VM_START_VNC_SERVER = 102,
    PJOC_VM_STOP_VNC_SERVER = 103,
    PJOC_VM_GUEST_GET_NETWORK_SETTINGS = 104,
    PJOC_SRV_GET_VIRTUAL_NETWORK_LIST = 105,
    PJOC_VM_AUTH_WITH_GUEST_SECURITY_DB = 106,
    PJOC_VM_GUEST_SET_USER_PASSWD = 107,
    PJOC_SRV_ADD_VIRTUAL_NETWORK = 108,
    PJOC_SRV_UPDATE_VIRTUAL_NETWORK = 109,
```

```
    PJOC_SRV_DELETE_VIRTUAL_NETWORK = 110,
    PJOC_SRV_CONFIGURE_GENERIC_PCI = 111,
    PJOC_SRV_PREPARE_FOR_HIBERNATE = 112,
    PJOC_SRV_AFTER_HOST_RESUME = 113,
    PJOC_SRV_GET_BACKUP_TREE = 114,
    PJOC_SRV_CREATE_VM_BACKUP = 115,
    PJOC_SRV_RESTORE_VM_BACKUP = 116,
    PJOC_API_SEND_PROBLEM_REPORT = 117,
    PJOC_VM_LOCK = 118,
    PJOC_VM_UNLOCK = 119,
    PJOC_SRV_SET_NON_INTERACTIVE_SESSION = 120,
    PJOC_SRV_CREATE_UNATTENDED_CD = 121,
    PJOC_SRV_REMOVE_VM_BACKUP = 122,
    PJOC_VM_RESIZE_DISK_IMAGE = 123,
    PJOC_SRV_SET_SESSION_CONFIRMATION_MODE = 124,
    PJOC_SRV_STORE_VALUE_BY_KEY = 125,
    PJOC_VM_STORE_VALUE_BY_KEY = 126,
    PJOC_VM_COMPACT = 127,
    PJOC_VM_CANCEL_COMPACT = 128,
    PJOC_SRV_SEND_CLIENT_STATISTICS = 129,
    PJOC_VM_GET_PACKED_PROBLEM_REPORT = 130,
    PJOC_SRV_GET_PACKED_PROBLEM_REPORT = 131,
    PJOC_API_SEND_PACKED_PROBLEM_REPORT = 132,
    PJOC_SRV_UPDATE_USB_ASSOC_LIST = 133,
    PJOC_SRV_REGISTER_3RD_PARTY_VM = 134,
    PJOC_VM_CHANGE_SID = 135,
    PJOC_VM_RESET_UPTIME = 136,
    PJOC_REPORT_ASSEMBLY = 137,
    PJOC_SRV_INSTALL_APPLIANCE = 138,
    PJOC_SRV_UPDATE_OFFLINE_SERVICE = 139,
    PJOC_SRV_DELETE_OFFLINE_SERVICE = 140,
    PJOC_SRV_GET_OFFLINE_SERVICES_LIST = 141,
    PJOC_SRV_START_CLUSTER_SERVICE = 142,
    PJOC_SRV_STOP_CLUSTER_SERVICE = 143,
    PJOC_VM_CMD_INTERNAL = 144,
    PJOC_VM_GET_VIRTUAL_DEVICES_INFO = 145,
    PJOC_VM_CONVERT_DISKS = 146,
    PJOC_SRV_UPDATE_NETWORK_CLASSES_CONFIG = 152,
    PJOC_SRV_GET_NETWORK_CLASSES_LIST = 153,
    PJOC_SRV_UPDATE_NETWORK_SHAPING_CONFIG = 154,
    PJOC_SRV_GET_NETWORK_SHAPING_CONFIG = 155,
    PJOC_SRV_RESTART_NETWORK_SHAPING = 156,
    PJOC_SRV_PROXY_GET_REGISTERED_HOSTS = 157,
    PJOC_SRV_REGISTER_ISCSI_STORAGE = 158,
    PJOC_SRV_UNREGISTER_ISCSI_STORAGE = 159,
    PJOC_SRV_EXTEND_ISCSI_STORAGE = 160,
    PJOC_SRV_GET_CT_TEMPLATE_LIST = 161,
    PJOC_SRV_CHECK_ALIVE = 162,
    PJOC_SRV_GET_DEFAULT_VM_CONFIG = 163,
    PJOC_SRV_REMOVE_CT_TEMPLATE = 164,
    PJOC_SRV_COPY_CT_TEMPLATE = 165,
    PJOC_SRV_ADD_IPPRIVATE_NETWORK = 166,
    PJOC_SRV_REMOVE_IPPRIVATE_NETWORK = 167,
    PJOC_SRV_UPDATE_IPPRIVATE_NETWORK = 168,
    PJOC_SRV_GET_IPPRIVATE_NETWORKS_LIST = 169,
    PJOC_VM_DEV_COPY_IMAGE = 170,
    PJOC_SRV_REFRESH_PLUGINS = 171,
    PJOC_VM_MOUNT = 172,
    PJOC_VM_UMOUNT = 173,
    PJOC_SRV_GET_PLUGINS_LIST = 174,
    PJOC_SRV_GET_DISK_FREE_SPACE = 175,
    PJOC_DESKTOP_CONTROL_CONNECT = 176,
    PJOC_VM_MOVE = 177,
    PJOC_SRV_GENERATE_CSR = 178,
    PJOC_SRV_SIGN_CERTIFICATION_REQUEST = 179,
    PJOC_SRV_GET_VM_CONFIG = 180,
```

```
    PJOC_PTM_RPC = 181,
    PJOC_API_PUT_USER_DATA = 182,
    PJOC_API_GET_USER_DATA = 183,
    PJOC_API_GET_USER_KEYS = 184,
    PJOC_API_REMOVE_USER_DATA = 185,
    PJOC_API_PUT_ICON = 186,
    PJOC_API_VALIDATE_IN_CACHE = 187,
    PJOC_SRV_UPDATE_SESSION_INFO = 188,
    PJOC_API_SUBSCRIBE_ON_REGISTRATION = 189,
    PJOC_SRV_SEND_PROBLEM_REPORT = 190,
    PJOC_VM_SEND_PROBLEM_REPORT = 191,
    PJOC_VM_SET_PROTECTION = 192,
    PJOC_VM_REMOVE_PROTECTION = 193,
    PJOC_SRV_CPU_POOLS_LIST_POOLS = 194,
    PJOC_SRV_CPU_POOLS_MOVE = 195,
    PJOC_SRV_CPU_POOLS_RECALCULATE = 196,
    PJOC_SRV_CPU_POOLS_JOIN = 197,
    PJOC_SRV_CPU_POOLS_LEAVE = 198,
    PJOC_VM_COMMIT_ENCRYPTION = 199,
    PJOC_SRV_GET_VCMMD_CONFIG = 200,
    PJOC_CT_REINSTALL = 201
} PRL_JOB_OPERATION_CODE;
```

File

PrlEnums.h

PRL_JOB_STATUS Enumeration

Job status enumeration

Syntax

```
typedef enum _PRL_JOB_STATUS {
  PJS_UNKNOWN = 0,
  PJS_RUNNING = 0xD0000001,
  PJS_FINISHED = 0xD0000002
} PRL_JOB_STATUS;
```

File

PrlEnums.h

PRL_KEY Enumeration

Virtuozzo virtual keycodes.

Syntax

```
typedef enum _PRL_KEY {
  PRL_KEY_INVALID = 0,
  PRL_KEY_WILDCARD_ANY = 1,
  PRL_KEY_WILDCARD_KEYBOARD = 2,
  PRL_KEY_WILDCARD_MOUSE = 3,
  PRL_KEY_WILDCARD_SHIFT = 4,
  PRL_KEY_WILDCARD_CTRL = 5,
  PRL_KEY_WILDCARD_ALT = 6,
  PRL_KEY_WILDCARD_WIN = 7,
  PRL_KEY_ESCAPE = 9,
  PRL_KEY_1 = 10,
```

```
    PRL_KEY_2 = 11,
    PRL_KEY_3 = 12,
    PRL_KEY_4 = 13,
    PRL_KEY_5 = 14,
    PRL_KEY_6 = 15,
    PRL_KEY_7 = 16,
    PRL_KEY_8 = 17,
    PRL_KEY_9 = 18,
    PRL_KEY_0 = 19,
    PRL_KEY_MINUS = 20,
    PRL_KEY_EQUAL = 21,
    PRL_KEY_BACKSPACE = 22,
    PRL_KEY_TAB = 23,
    PRL_KEY_Q = 24,
    PRL_KEY_W = 25,
    PRL_KEY_E = 26,
    PRL_KEY_R = 27,
    PRL_KEY_T = 28,
    PRL_KEY_Y = 29,
    PRL_KEY_U = 30,
    PRL_KEY_I = 31,
    PRL_KEY_O = 32,
    PRL_KEY_P = 33,
    PRL_KEY_LEFT_BRACKET = 34,
    PRL_KEY_RIGHT_BRACKET = 35,
    PRL_KEY_ENTER = 36,
    PRL_KEY_LEFT_CONTROL = 37,
    PRL_KEY_A = 38,
    PRL_KEY_S = 39,
    PRL_KEY_D = 40,
    PRL_KEY_F = 41,
    PRL_KEY_G = 42,
    PRL_KEY_H = 43,
    PRL_KEY_J = 44,
    PRL_KEY_K = 45,
    PRL_KEY_L = 46,
    PRL_KEY_SEMICOLON = 47,
    PRL_KEY_QUOTE = 48,
    PRL_KEY_TILDA = 49,
    PRL_KEY_LEFT_SHIFT = 50,
    PRL_KEY_BACKSLASH = 51,
    PRL_KEY_Z = 52,
    PRL_KEY_X = 53,
    PRL_KEY_C = 54,
    PRL_KEY_V = 55,
    PRL_KEY_B = 56,
    PRL_KEY_N = 57,
    PRL_KEY_M = 58,
    PRL_KEY_COMMA = 59,
    PRL_KEY_DOT = 60,
    PRL_KEY_SLASH = 61,
    PRL_KEY_RIGHT_SHIFT = 62,
    PRL_KEY_NP_STAR = 63,
    PRL_KEY_LEFT_ALT = 64,
    PRL_KEY_SPACE = 65,
    PRL_KEY_CAPS_LOCK = 66,
    PRL_KEY_F1 = 67,
    PRL_KEY_F2 = 68,
    PRL_KEY_F3 = 69,
    PRL_KEY_F4 = 70,
    PRL_KEY_F5 = 71,
    PRL_KEY_F6 = 72,
    PRL_KEY_F7 = 73,
    PRL_KEY_F8 = 74,
    PRL_KEY_F9 = 75,
    PRL_KEY_F10 = 76,
```

```
    PRL_KEY_NUM_LOCK = 77,
    PRL_KEY_SCROLL_LOCK = 78,
    PRL_KEY_NP_7 = 79,
    PRL_KEY_NP_8 = 80,
    PRL_KEY_NP_9 = 81,
    PRL_KEY_NP_MINUS = 82,
    PRL_KEY_NP_4 = 83,
    PRL_KEY_NP_5 = 84,
    PRL_KEY_NP_6 = 85,
    PRL_KEY_NP_PLUS = 86,
    PRL_KEY_NP_1 = 87,
    PRL_KEY_NP_2 = 88,
    PRL_KEY_NP_3 = 89,
    PRL_KEY_NP_0 = 90,
    PRL_KEY_NP_DELETE = 91,
    PRL_KEY_PRINT = 92,
    PRL_KEY_EUROPE_1 = 93,
    PRL_KEY_EUROPE_2 = 94,
    PRL_KEY_F11 = 95,
    PRL_KEY_F12 = 96,
    PRL_KEY_HOME = 97,
    PRL_KEY_UP = 98,
    PRL_KEY_PAGE_UP = 99,
    PRL_KEY_LEFT = 100,
    PRL_KEY_RIGHT = 102,
    PRL_KEY_END = 103,
    PRL_KEY_DOWN = 104,
    PRL_KEY_PAGE_DOWN = 105,
    PRL_KEY_INSERT = 106,
    PRL_KEY_DELETE = 107,
    PRL_KEY_NP_ENTER = 108,
    PRL_KEY_RIGHT_CONTROL = 109,
    PRL_KEY_PAUSE = 110,
    PRL_KEY_NP_SLASH = 112,
    PRL_KEY_RIGHT_ALT = 113,
    PRL_KEY_LEFT_WIN = 115,
    PRL_KEY_RIGHT_WIN = 116,
    PRL_KEY_MENU = 117,
    PRL_KEY_MEDIA_NEXT_TRACK = 118,
    PRL_KEY_MEDIA_PREV_TRACK = 119,
    PRL_KEY_MEDIA_STOP = 120,
    PRL_KEY_MEDIA_PLAY_PAUSE = 121,
    PRL_KEY_MUTE = 122,
    PRL_KEY_VOLUME_UP = 123,
    PRL_KEY_VOLUME_DOWN = 124,
    PRL_KEY_MEDIA_SELECT = 125,
    PRL_KEY_APP_MAIL = 126,
    PRL_KEY_APP_CALCULATOR = 127,
    PRL_KEY_APP_MY_COMPUTER = 128,
    PRL_KEY_WWW_SEARCH = 129,
    PRL_KEY_WWW_HOME = 130,
    PRL_KEY_WWW_BACK = 131,
    PRL_KEY_WWW_FORWARD = 132,
    PRL_KEY_WWW_STOP = 133,
    PRL_KEY_WWW_REFRESH = 134,
    PRL_KEY_WWW_FAVORITES = 135,
    PRL_KEY_EJECT = 136,
    PRL_KEY_SYSTEM_POWER = 137,
    PRL_KEY_SYSTEM_SLEEP = 138,
    PRL_KEY_SYSTEM_WAKE = 139,
    PRL_KEY_BRAZILIAN_KEYPAD = 140,
    PRL_KEY_RO = 141,
    PRL_KEY_HIRAGANA_KATAKANA = 142,
    PRL_KEY_YEN = 143,
    PRL_KEY_HENKAN = 144,
    PRL_KEY_MUHENKAN = 145,
```

```
    PRL_KEY_PC9800_KEYPAD = 146,
    PRL_KEY_HANGUEL = 147,
    PRL_KEY_HANJA = 148,
    PRL_KEY_KATAKANA = 149,
    PRL_KEY_HIRAGANA = 150,
    PRL_KEY_ZENKAKU_HANKAKU = 151,
    PRL_KEY_F13 = 152,
    PRL_KEY_F14 = 153,
    PRL_KEY_F15 = 154,
    PRL_KEY_F16 = 155,
    PRL_KEY_F17 = 156,
    PRL_KEY_F18 = 157,
    PRL_KEY_F19 = 158,
    PRL_KEY_F20 = 159,
    PRL_KEY_F21 = 160,
    PRL_KEY_F22 = 161,
    PRL_KEY_F23 = 162,
    PRL_KEY_F24 = 163,
    PRL_KEY_NP_EQUAL = 164,
    PRL_KEY_BREAK = 165,
    PRL_KEY_PRINT_WITH_MODIFIER = 166,
    PRL_KEY_SYSRQ = 167,
    PRL_KEY_FN = 168,
    PRL_KEY_EURO = 169,
    PRL_KEY_DOLLAR = 170,
    PRL_KEY_MON_BRIGHTNESS_DOWN = 171,
    PRL_KEY_MON_BRIGHTNESS_UP = 172,
    PRL_KEY_APP_EXPOSE = 173,
    PRL_KEY_APP_DASHBOARD = 174,
    PRL_KEY_KBD_BRIGHTNESS_DOWN = 175,
    PRL_KEY_KBD_BRIGHTNESS_UP = 176,
    PRL_KEY_MAC129 = 177,
    PRL_KEY_LEFT_BUTTON = 178,
    PRL_KEY_MIDDLE_BUTTON = 179,
    PRL_KEY_RIGHT_BUTTON = 180,
    PRL_KEY_MOVE_UP_LEFT = 181,
    PRL_KEY_MOVE_UP = 182,
    PRL_KEY_MOVE_UP_RIGHT = 183,
    PRL_KEY_MOVE_LEFT = 184,
    PRL_KEY_MOVE_RIGHT = 185,
    PRL_KEY_MOVE_DOWN_LEFT = 186,
    PRL_KEY_MOVE_DOWN = 187,
    PRL_KEY_MOVE_DOWN_RIGHT = 188,
    PRL_KEY_WHEEL_UP = 189,
    PRL_KEY_WHEEL_DOWN = 190,
    PRL_KEY_WHEEL_LEFT = 191,
    PRL_KEY_WHEEL_RIGHT = 192,
    PRL_KEY_MAX = 193
  } PRL_KEY;
```

File

PrlKeys.h

PRL_KEY_EVENT Enumeration

Syntax

```
  typedef enum _PRL_KEY_EVENT {
    PKE_PRESS = 0x0,
    PKE_RELEASE = 0x80
  } PRL_KEY_EVENT;
```

File

PrlEnums.h

PRL_LICENSE_RESTRICTION_KEY Enumeration

License restriction options

Syntax

```
typedef enum _PRL_LICENSE_RESTRICTION_KEY {
  PLRK_UNKNOWN = 0,
  PLRK_VM_CPU_LIMIT = 1,
  PLRK_VM_MEMORY_LIMIT = 2,
  PLRK_VM_VTD_AVAILABLE = 3,
  PLRK_ALLOWED_GUEST_OS_VERSIONS = 4,
  PLRK_RUNNING_VMS_LIMIT = 5,
  PLRK_VM_CREATE = 6,
  PLRK_VM_REGISTER = 7,
  PLRK_VM_IMPORT_3RD_PARTY = 8,
  PLRK_VM_CLONE = 9,
  PLRK_VM_CONVERT_TO_TEMPLATE = 10,
  PLRK_VM_CONVERT_FROM_TEMPLATE = 11,
  PLRK_VM_DOWNLOAD = 12,
  PLRK_VM_SUSPEND = 13,
  PLRK_VM_PAUSE = 14,
  PLRK_VM_SNAPSHOT_CREATE = 15,
  PLRK_VM_SNAPSHOT_SWITCH = 16,
  PLRK_VM_SNAPSHOT_DELETE = 17,
  PLRK_VM_SNAPSHOT_TREE = 18,
  PLRK_VM_UNDODISK_FEATURE = 19,
  PLRK_VM_SAFEMODE_FEATURE = 20,
  PLRK_VM_SMARTGUARD_FEATURE = 21,
  PLRK_VM_SHOW_FULLSCREEN = 22,
  PLRK_RUNNING_VMS_LIMIT_PER_USER = 23
} PRL_LICENSE_RESTRICTION_KEY;
```

File

PrlEnums.h

Members

PLRK_UNKNOWN

NOTE to Virtuozzo Developers:

- DO NOT CHANGE ANY MEMBER NAME BECAUSE IT USES IN INTERNAL PROTOCOL(ParallelsDomModel.h)

PLRK_VM_CPU_LIMIT

Value is PRL_UINT32 (p. 798)

PLRK_VM_MEMORY_LIMIT

Value is PRL_UINT32 (p. 798) in Mb

PLRK_VM_VTD_AVAILABLE

Value is PRL_UINT32 (p. 798)

PLRK_ALLOWED_GUEST_OS_VERSIONS

Value is PHT_OPAQUE_TYPE_LIST (p. 177) of Os version values from PrlOses.h

PLRK_RUNNING_VMS_LIMIT

Value is PRL_UINT32 (p. 798) (max count on the host)

PLRK_VM_CREATE
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_REGISTER
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_IMPORT_3RD_PARTY
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_CLONE
Value is PRL_UINT32 (p. 798) - max count of Vm in system after this operation

PLRK_VM_CONVERT_TO_TEMPLATE
Value is PRL_UINT32 (p. 798)

PLRK_VM_CONVERT_FROM_TEMPLATE
Value is PRL_UINT32 (p. 798)

PLRK_VM_DOWNLOAD
Value is PRL_UINT32 (p. 798)

PLRK_VM_SUSPEND
Value is PRL_UINT32 (p. 798)

PLRK_VM_PAUSE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_CREATE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_SWITCH
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_DELETE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SNAPSHOT_TREE
Value is PRL_UINT32 (p. 798)

PLRK_VM_UNDODISK_FEATURE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SAFEMODE_FEATURE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SMARTGUARD_FEATURE
Value is PRL_UINT32 (p. 798)

PLRK_VM_SHOW_FULLSCREEN
Value is PRL_UINT32 (p. 798)

PLRK_RUNNING_VMS_LIMIT_PER_USER
Value is PRL_UINT32 (p. 798) Look also at PLRK_RUNNING_VMS_LIMIT

PRL_LOGIN_LOCAL_FLAGS Enumeration

PrlSrv_LoginLocal (p. 276) flags

Syntax

```
typedef enum _PRL_LOGIN_LOCAL_FLAGS {
```

```
  PLLF_LOGIN_WITH_DELAYED_CONNECTION = 1<<(PACF_MAX+1)
} PRL_LOGIN_LOCAL_FLAGS;
```

File

PrlCommandsFlags.h

PRL_MASS_STORAGE_INTERFACE_TYPE Enumeration

Mass storage device interface type

Syntax

```
typedef enum _PRL_MASS_STORAGE_INTERFACE_TYPE {
  PMS_IDE_DEVICE = 0,
  PMS_SCSI_DEVICE = 1,
  PMS_SATA_DEVICE = 2,
  PMS_VIRTIO_BLOCK_DEVICE = 3,
  PMS_UNKNOWN_DEVICE = 255
} PRL_MASS_STORAGE_INTERFACE_TYPE;
```

File

PrlEnums.h

PRL_MEMGUARANTEE_TYPE Enumeration

Syntax

```
typedef enum _PRL_MEMGUARANTEE_TYPE {
  PRL_MEMGUARANTEE_AUTO = 0,
  PRL_MEMGUARANTEE_PERCENTS = 1
} PRL_MEMGUARANTEE_TYPE;
```

File

PrlEnums.h

PRL_MESSAGE_TYPE_ENUM Enumeration

Message type enumeration.

Syntax

```
typedef enum _PRL_MESSAGE_TYPE_ENUM {
  PMT_WARNING = 0,
  PMT_INFORMATION = 1,
  PMT_CRITICAL = 2,
  PMT_QUESTION = 3,
  PMT_ANSWER = 4
} PRL_MESSAGE_TYPE_ENUM;
```

File

PrlEnums.h

PRL_MOBILE_ADVANCED_AUTH_MODE Enumeration

Authorization mode, when mobile client establishing remote access to the host.

Syntax

```
typedef enum _PRL_MOBILE_ADVANCED_AUTH_MODE {
  PMAA_NO_ADVANCED_AUTH_NEEDED = 0,
  PMAA_USE_SYSTEM_CREDENTIALS = 1
} PRL_MOBILE_ADVANCED_AUTH_MODE;
```

File

PrlEnums.h

PRL_MOUNT_VIRTUAL_DISK Enumeration

Mount virtual disk flags

Syntax

```
typedef enum _PRL_MOUNT_VIRTUAL_DISK {
  PMVD_READ_ONLY = 1<<(PACF_MAX+1),
  PMVD_READ_WRITE = 1<<(PACF_MAX+2),
  PMVD_INFO = 1<<(PACF_MAX+3)
} PRL_MOUNT_VIRTUAL_DISK;
```

File

PrlCommandsFlags.h

PRL_MOUSE_BUTTON Enumeration

Syntax

```
typedef enum _PRL_MOUSE_BUTTON {
  PMB_NOBUTTON = 0,
  PMB_LEFT_BUTTON = 1<<0,
  PMB_RIGHT_BUTTON = 1<<1,
  PMB_MIDDLE_BUTTON = 1<<2,
  PMB_XBUTTON1 = 1<<3,
  PMB_XBUTTON2 = 1<<4,
  PMB_XBUTTON3 = 1<<5,
  PMB_XBUTTON4 = 1<<6,
  PMB_XBUTTON5 = 1<<7
} PRL_MOUSE_BUTTON;
```

File

PrlEnums.h

PRL_NET_ADAPTER_EMULATED_TYPE Enumeration

Network adapter emulated type

Syntax

```
typedef enum _PRL_NET_ADAPTER_EMULATED_TYPE {
  PNA_HOST_ONLY = 0,
  PNA_SHARED = 1,
  PNA_BRIDGED_NETWORK = 2,
  PNA_BRIDGED_ETHERNET = PNA_BRIDGED_NETWORK,
  PNA_DIRECT_ASSIGN = 4,
  PNA_ROUTED = 5,
  PNA_BRIDGE = 6
} PRL_NET_ADAPTER_EMULATED_TYPE;
```

File

PrlEnums.h

PRL_NET_ADAPTER_TYPE Enumeration

Syntax

```
typedef enum _PRL_NET_ADAPTER_TYPE {
  NIC_TYPE_PHYSICAL = 0,
  NIC_TYPE_BRIDGE = 1,
  NIC_TYPE_VLAN = 2,
  NIC_TYPE_VETH = 3
} PRL_NET_ADAPTER_TYPE;
```

File

PrlEnums.h

PRL_NET_VIRTUAL_NETWORK_TYPE Enumeration

Virtual Network type.

Syntax

```
typedef enum _PRL_NET_VIRTUAL_NETWORK_TYPE {
  PVN_BRIDGED_ETHERNET = 0,
  PVN_HOST_ONLY = 1
} PRL_NET_VIRTUAL_NETWORK_TYPE;
```

File

PrlEnums.h

Remarks

Should match values in LibrariesPrlNetworkingprl_netconfig.h

PRL_NETFILTER_MODE Enumeration

Netfilter modes list

Syntax

```
typedef enum _PRL_NETFILTER_MODE {
  PCNM_NOT_SET = 0,
  PCNM_DISABLED = 1,
  PCNM_STATELESS = 2,
  PCNM_STATEFUL = 3,
  PCNM_FULL = 4
} PRL_NETFILTER_MODE;
```

File

PrlEnums.h

PRL_NETWORK_RATE_MPU Enumeration

Special values of network packet MPU for packet rate limitation.

Syntax

```
typedef enum _PRL_NETWORK_RATE_MPU {
  NRM_DISABLED = -1,
  NRM_ENABLED = 0
} PRL_NETWORK_RATE_MPU;
```

File

PrlEnums.h

Members

NRM_ENABLED
default MPU

PRL_ONLINE_COMPACT_MODE Enumeration

Online compact mode for virtual machine disks

Syntax

```
typedef enum _PRL_ONLINE_COMPACT_MODE {
  POCM_DISABLED = 0,
  POCM_ENABLED = 1,
  POCM_AUTO = 2
} PRL_ONLINE_COMPACT_MODE;
```

File

PrlEnums.h

PRL_OPTIMIZE_MODIFIERS_MODE Enumeration

Determines the mode of modifiers handling inside the guest

Syntax

```
typedef enum _PRL_OPTIMIZE_MODIFIERS_MODE {
  POD_STANDARD = 0,
  POD_OPTIMIZE_FOR_GAMES = 1,
  POD_OPTIMIZE_FOR_ACCESSIBILITY = 2
} PRL_OPTIMIZE_MODIFIERS_MODE;
```

File

PrlEnums.h

PRL_PARAM_FIELD_DATA_TYPE Enumeration

Event parameters fields types

Syntax

```
typedef enum _PRL_PARAM_FIELD_DATA_TYPE {
  PFD_UINT32 = 0,
  PFD_STRING = 1,
  PFD_INT32 = 2,
  PFD_CDATA = 3,
  PFD_BOOLEAN = 4,
  PFD_ENTITY = 5,
  PFD_ENUMERATION = 6,
  PFD_INT64 = 7,
  PFD_UINT64 = 8,
  PFD_BINARY = 9,
  PFD_UNKNOWN = 0xFF
} PRL_PARAM_FIELD_DATA_TYPE;
```

File

PrlEnums.h

PRL_PORT_FORWARDING_TYPE Enumeration

Port forwarding type: PPF_TCP : TCP port forwarding PPF_UDP : UDP port forwarding

Syntax

```
typedef enum _PRL_PORT_FORWARDING_TYPE {
  PPF_TCP = 0,
  PPF_UDP = 1
} PRL_PORT_FORWARDING_TYPE;
```

File

PrlEnums.h

PRL_PRINTER_INTERFACE_TYPE Enumeration

Printer device interface type

Syntax

```
typedef enum _PRL_PRINTER_INTERFACE_TYPE {
  PRN_LPT_DEVICE = 0,
  PRN_USB_DEVICE = 1,
  PRN_UNKNOWN_DEVICE = 255
} PRL_PRINTER_INTERFACE_TYPE;
```

File

PrlEnums.h

PRL_PRIVATE_EVENT_TYPE Enumeration

Private event types.

Syntax

```
typedef enum _PRL_PRIVATE_EVENT_TYPE {
  PET_IO_DISPLAY_PALETTE_CHANGED = 200000,
  PET_IO_TOOLS_TIS_QUERY = 200001,
  PET_IO_SCREEN_UPDATED = 200002,
  PET_IO_AUTH_RESPONSE = 200003,
  PET_IO_ATTACH_RESPONSE = 200004,
  PET_IO_ENCODING_RESPONSE = 200005,
  PET_IO_MEMORY_INFO = 200006,
  PET_IO_SCREEN_CAPTURED_BUFFER = 200007,
  PET_IO_TOOLS_TIS_SUBSCRIBE = 200008,
  PET_IO_TOOLS_TIS_UNSUBSCRIBE = 200009,
  PET_IO_CLI_AUTHENTICATE_SESSION = 200100,
  PET_IO_CLI_ATTACH_TO_VM = 200101,
  PET_IO_CLI_ENCODING_REQUEST = 200102,
  PET_IO_CLI_SCREEN_WATCH_UPDATES = 200103,
  PET_IO_CLI_SCREEN_CAPTURE_BUFFER = 200104,
  PET_IO_CLI_SCREEN_SIZE = 200105,
  PET_IO_CLI_KEYBOARD_SCANCODE = 200106,
  PET_IO_CLI_MOUSE_SETPOS = 200107,
  PET_IO_CLI_MOUSE_MOVE = 200108,
  PET_IO_CLI_MOUSE_NEED_CURSOR = 200109,
  PET_IO_CLI_GRACEFUL_SHUTDOWN = 200110,
  PET_IO_CLI_SCREEN_SET_SURFACE = 200111,
  PET_IO_CLI_REMOTE_COMMAND = 200112,
  PET_IO_CLI_TOOLS_UTILITY_COMMAND = 200113,
  PET_IO_CLI_TOOLS_TIS_QUERY = 200114,
  PET_IO_CLI_TOOLS_CLIPBOARD_DATA = 200115,
  PET_IO_CLI_TOOLS_SIA_DATA = 200116,
  PET_IO_CLI_TOOLS_VMCTG_COMMAND = 200117,
  PET_IO_CLI_TOOLS_GENERAL_COMMAND = 200118,
  PET_IO_CLI_TOOLS_EMPTY_RECYCLE_BIN = 200119,
  PET_IO_CLI_TOOLS_SET_TASKBAR_VISIBILITY = 200120,
  PET_IO_CLI_TOOLS_TOGGLE_DESKTOP = 200121,
  PET_IO_CLI_TOOLS_MOVE_TASKBAR = 200122,
  PET_IO_CLI_TOOLS_SHOW_RECYCLE_BIN = 200123,
  PET_IO_CLI_TOOLS_NOTIFY_COHERENCE_STATE = 200124,
  PET_IO_CLI_TOOLS_ADJUST_WORKING_AREA = 200125,
```

```
    PET_IO_CLI_KEYBOARD_KEY = 200126,
    PET_IO_CLI_TOOLS_UIEMU = 200127,
    DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_STREAM_START = 200128,
    DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_STREAM_STOP = 200129,
    DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_SET_ENCODING = 200130,
    PET_IO_CLI_CVSRC_CONNECT = 200131,
    PET_IO_CLI_CVSRC_DISCONNECT = 200132,
    PET_IO_CLI_AUTHENTICATE_EXEC_SESSION = 200133,
    PET_IO_CLI_SWITCH_SLIDING_MOUSE = 200134,
    PET_IO_CLI_SSO_SET_CREDENTIALS = 200135,
    PET_IO_CLI_SSO_SET_CREDENTIALS_ASYNC = 200136,
    PET_IO_CLI_SSO_LOCKED = 200137,
    PET_IO_CLI_SSO_UNLOCKED = 200138,
    PET_IO_CLI_DPI_SCREEN_SIZE = 200139,
    PET_IO_CLI_TOOLS_SET_INDENTS = 200140,
    PET_IO_CLI_AUDIO_RECORDING_STREAM_START = 200141,
    PET_IO_CLI_AUDIO_RECORDING_STREAM_STOP = 200142,
    PET_IO_CLI_AUDIO_PLAYOUT_STREAM_START = 200143,
    PET_IO_CLI_AUDIO_PLAYOUT_STREAM_STOP = 200144,
    PET_IO_CLI_TOOLS_OPEN_FILE_BROWSER = 200145,
    PET_IO_STDIN_PORTION = 200200,
    PET_IO_STDOUT_PORTION = 200201,
    PET_IO_STDERR_PORTION = 200202,
    PET_IO_READY_TO_ACCEPT_STDOUT_STDERR_PKGS = 200203,
    PET_IO_FIN_TO_TRANSMIT_STDOUT_STDERR = 200204,
    PET_IO_CLIENT_PROCESSED_ALL_DESCS_DATA = 200205,
    PET_IO_TTY_SETTINGS_REQUIRED = 200206,
    PET_IO_TTY_SETTINGS_PROVIDED = 200207,
    PET_IO_STDIN_WAS_CLOSED = 200208,
    PET_IT_RECONFIG_BOOTCAMP_START = 200300,
    PET_IT_VALIDATE_BOOTCAMP_START = 200301,
    PET_IT_CONVERT_TO_OLD_START = 200302,
    PET_IT_CONVERT_TO_CURRENT_START = 200303,
    PET_IT_RECONFIG_START = 200304,
    PET_IT_MERGE_SNAPSHOTS_START = 200305,
    PET_IT_MAKE_PLAIN_START = 200306,
    PET_IT_MAKE_EXPANDING_START = 200307,
    PET_IT_MAKE_SPLIT_START = 200308,
    PET_IT_MAKE_NON_SPLIT_START = 200309,
    PET_IT_GET_RESIZE_INFORMATION_START = 200310,
    PET_IT_INCREASE_LAST_VOLUME_START = 200311,
    PET_IT_RESIZING_START = 200312,
    PET_IT_DECREASING_START = 200313,
    PET_IT_COMPACTING_START = 200314,
    PET_IT_FIX_CONSISTENCY_START = 200315,
    PET_IT_CONVERT_TO_EXTEND_START = 200316,
    PET_DESKCTL_SESSION_AUTH = 200400,
    PET_DESKCTL_SESSION_AUTH_FIN = 200401,
    PET_DESKCTL_ATTACH = 200402,
    PET_DESKCTL_ATTACH_FIN = 200403,
    PET_DESKCTL_LOCAL_AUTH = 200404,
    PET_DESKCTL_LOCAL_AUTH_QUEST = 200405,
    PET_DESKCTL_LOCAL_AUTH_ANSWER = 200406,
    PET_DESKCTL_LOCAL_AUTH_FIN = 200407,
    PET_DESKCTL_ACCEPT_CLIENTS = 200408,
    PET_DESKCTL_REJECT_CLIENTS = 200409,
    PET_DESKCTL_CONFIG = 200410,
    PET_DESKCTL_CONFIG_QUERY = 200411,
    PET_DESKCTL_SEND_SAS = 200412,
    PET_DESKCTL_STATUS_CHANGE = 200413,
    PET_IO_DISCONNECT_WITH_REASON = 200500,
    PET_IO_DISPLAY_RESOLUTION_SET = 200510,
    PET_VIDEO_STREAM_START = 200600,
    PET_VIDEO_STREAM_CLOSE = 200601,
    PET_VIDEO_STREAM_FORMAT = 200602,
    PET_VIDEO_STREAM_DATA = 200603,
```

```
    PET_VIDEO_STREAM_CLOSED = 200604,
    PET_VIDEO_STREAM_ACK_FRAME = 200605,
    PET_VIDEO_STREAM_OPEN = 200606,
    PET_VIDEO_DATA_CHANNEL_CTL = 200607,
    PET_VIDEO_STREAM_STOP = 200608,
    PET_IO_PTM_PROTOCOL = 200700,
    PET_IO_IOSERVICE_CHANNELS_PROTOCOL = 200800
  } PRL_PRIVATE_EVENT_TYPE;
```

File

PrlEventsValues.h

Members

PET_IO_DISPLAY_PALETTE_CHANGED
BASE VALUE SET TO 200000

PET_IO_CLI_AUTHENTICATE_SESSION
ALIGNED VALUE TO 200100

PET_IO_STDIN_PORTION
ALIGNED VALUE TO 200200

PET_IT_RECONFIG_BOOTCAMP_START
ALIGNED VALUE TO 200300

PET_DESKCTL_SESSION_AUTH
ALIGNED VALUE TO 200400

PET_IO_DISCONNECT_WITH_REASON
ALIGNED VALUE TO 200500

PET_IO_DISPLAY_RESOLUTION_SET
ALIGNED VALUE TO 200510

PET_VIDEO_STREAM_START
ALIGNED VALUE TO 200600

PET_IO_PTM_PROTOCOL
ALIGNED VALUE TO 200700

PET_IO_IOSERVICE_CHANNELS_PROTOCOL
ALIGNED VALUE TO 200800

Remarks

Is NOT a part of public API.

PRL_PROBLEM_REPORT_FLAGS Enumeration

Virtuozzo problem report flags functionality: PPRF_ADD_CLIENT_PART : specifies to add client information to problem report PPRF_ADD_SERVER_PART : specifies to add server information to problem report PPRF_DO_NOT_CREATE_HOST_SCREENSHOT : specifies do not create and add host screenshot to generating problem report

Syntax

```
typedef enum _PRL_PROBLEM_REPORT_FLAGS {
  PPRF_ADD_CLIENT_PART = 1<<(PACF_MAX+1),
  PPRF_ADD_SERVER_PART = 1<<(PACF_MAX+2),
  PPRF_DO_NOT_CREATE_HOST_SCREENSHOT = 1<<(PACF_MAX+3)
```

```
} PRL_PROBLEM_REPORT_FLAGS;
```

File

PrlCommandsFlags.h

PRL_PROBLEM_REPORT_REASON Enumeration

Problem report reason enumeration

Syntax

```
typedef enum _PRL_PROBLEM_REPORT_REASON {
  PRR_UNKNOWN = 0,
  PRR_BOOT_CAMP = 1,
  PRR_HANG_LOCKUP = 2,
  PRR_KEYBOARD = 3,
  PRR_NETWORK = 4,
  PRR_PARALLELS_TOOLS = 5,
  PRR_PERFORMANCE = 6,
  PRR_PRODUCT_REGISTRATION = 7,
  PRR_SNAPSHOTS = 8,
  PRR_SOUND = 9,
  PRR_SUPPORT_REQUEST = 10,
  PRR_USB = 11,
  PRR_VIDEO_3D_GRAPHICS = 12,
  PRR_OTHER = 13,
  PRR_BACKUP = 14,
  PRR_MIGRATION = 15,
  PRR_CONNECTION = 16,
  PRR_LAST = PRR_CONNECTION
} PRL_PROBLEM_REPORT_REASON;
```

File

PrlEnums.h

Members

PRR_LAST
Should equals to the last item

PRL_PROBLEM_REPORT_SCHEME Enumeration

Problem report scheme enumaration

Syntax

```
typedef enum _PRL_PROBLEM_REPORT_SCHEME {
  PRS_OLD_XML_BASED = 0,
  PRS_NEW_PACKED = 1
} PRL_PROBLEM_REPORT_SCHEME;
```

File

PrlEnums.h

PRL_PROBLEM_REPORT_TYPE Enumeration

Problem report type enumeration

Syntax

```
typedef enum _PRL_PROBLEM_REPORT_TYPE {
  PRT_REPORT_TYPE_UNDEFINED = 0,
  PRT_AUTOMATIC_VM_GENERATED_REPORT = 1,
  PRT_AUTOMATIC_DISPATCHER_GENERATED_REPORT = 2,
  PRT_USER_DEFINED_ON_STOPPED_VM_REPORT = 3,
  PRT_USER_DEFINED_ON_RUNNING_VM_REPORT = 4,
  PRT_USER_DEFINED_ON_CONNECTED_SERVER = 5,
  PRT_USER_DEFINED_ON_DISCONNECTED_SERVER = 6,
  PRT_USER_DEFINED_ON_NOT_RESPONDING_VM_REPORT = 7,
  PRT_AUTOMATIC_DETECTED_REPORT = 8,
  PRT_AUTOMATIC_STATISTICS_REPORT = 9,
  PRT_AUTOMATIC_INSTALLED_SOFTWARE_REPORT = 10,
  PRT_AUTOMATIC_VZ_STATISTICS_REPORT = 11,
  PRT_AUTOMATIC_UTW7AGENT_PRE_MIGRATION = 12,
  PRT_AUTOMATIC_UTW7AGENT_POST_MIGRATION = 13,
  PRT_AUTOMATIC_TRANSPORTER_REPORT = 14,
  PRT_USER_DEFINED_TRANSPORTER_REPORT = 15,
  PRT_AUTOMATIC_GUEST_GENERATED_REPORT = 16,
  PRT_USER_DEFINED_ON_CONTAINER_REPORT = 17,
  PRT_USER_DEFINED_ON_MOBILE_REPORT = 18,
  PRT_USER_DEFINED_ON_MOBILE_FEEDBACK_REPORT = 19,
  PRT_LAST = PRT_USER_DEFINED_ON_MOBILE_FEEDBACK_REPORT
} PRL_PROBLEM_REPORT_TYPE;
```

File

PrlEnums.h

Members

PRT_LAST
Should equals to the last item

PRL_PROCESS_STATE_TYPE Enumeration

Process state element

Syntax

```
typedef enum _PRL_PROCESS_STATE_TYPE {
  PPS_PROC_SLEEP = 0,
  PPS_PROC_RUN = 1,
  PPS_PROC_STOP = 2,
  PPS_PROC_ZOMBIE = 3,
  PPS_PROC_IDLE = 4
} PRL_PROCESS_STATE_TYPE;
```

File

PrlEnums.h

PRL_PRODUCT_ENUM Enumeration

Virtuozzo license products enumeration.

Syntax

```
typedef enum _PRL_PRODUCT_ENUM {
  PRL_PRODUCT_ANY = 0,
  PRL_PRODUCT_DESKTOP = 1,
  PRL_PRODUCT_SERVER = 2,
  PRL_PRODUCT_WORKSTATION = 3,
  PRL_PRODUCT_STM = 4,
  PRL_PRODUCT_PLAYER = 5,
  PRL_PRODUCT_DESKTOP_WL = 6
} PRL_PRODUCT_ENUM;
```

File

PrlEnums.h

PRL_PROXY_CONNECTION_STATUS Enumeration

Proxy connection status

Syntax

```
typedef enum _PRL_PROXY_CONNECTION_STATUS {
  PCS_DISCONNECTED = 0,
  PCS_CONNECTED = 1,
  PCS_CONNECTING = 2
} PRL_PROXY_CONNECTION_STATUS;
```

File

PrlEnums.h

PRL_QUERY_ELEMENT_FLAGS Enumeration

Query element flags

Syntax

```
typedef enum _PRL_QUERY_ELEMENT_FLAGS {
  PRL_UIEMU_ELEMENT_SEARCH_AREA_WINDOW = 0,
  PRL_UIEMU_ELEMENT_SEARCH_AREA_GLOBAL = 1
} PRL_QUERY_ELEMENT_FLAGS;
```

File

PrlEnums.h

PRL_REGISTER_3PARTY_VM_FLAGS Enumeration

PrlSrv_Register3rdPartyVm (p. 285) command flags set

Syntax

```
typedef enum _PRL_REGISTER_3PARTY_VM_FLAGS {
  PR3F_ALLOW_UNKNOWN_OS = 1<<(PACF_MAX+1),
  PR3F_CREATE_SHADOW_VM = 1<<(PACF_MAX+2)
} PRL_REGISTER_3PARTY_VM_FLAGS;
```

File

PrlCommandsFlags.h

PRL_REGISTER_NEW_VM Enumeration

PrlVm_RegEx (p. 485) flags

Syntax

```
typedef enum _PRL_REGISTER_NEW_VM {
  PRNVM_CREATE_FROM_LION_RECOVERY_PARTITION = PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION,
  PRNVM_ALLOW_TO_AUTO_DECREASE_HDD_SIZE = 1<<(PACF_MAX+2),
  PRNVM_PRESERVE_DISK = 1<<(PACF_MAX+3)
} PRL_REGISTER_NEW_VM;
```

File

PrlCommandsFlags.h

PRL_REGISTER_VM_FLAGS Enumeration

PrlSrv_RegisterVmEx (p. 288) command flags set

Syntax

```
typedef enum _PRL_REGISTER_VM_FLAGS {
  PRVF_REGENERATE_VM_UUID = 1<<(PACF_MAX+1),
  PRVF_KEEP_OTHERS_PERMISSIONS = 1<<(PACF_MAX+2),
  PRCF_FORCE = 1<<(PACF_MAX+3),
  PRVF_REGENERATE_SRC_VM_UUID = 1<<(PACF_MAX+4),
  PRVF_IGNORE_HA_CLUSTER = 1<<(PACF_MAX+5),
  PRCF_UNREG_PRESERVE = 1<<(PACF_MAX+6)
} PRL_REGISTER_VM_FLAGS;
```

File

PrlCommandsFlags.h

PRL_RESIZE_IMAGE_FLAGS Enumeration

PrlVmDev_ResizeImage (p. 766) command flags set

Syntax

```
typedef enum _PRL_RESIZE_IMAGE_FLAGS {
  PRIF_RESIZE_LAST_PARTITION = 1<<(PACF_MAX+1),
  PRIF_DISK_INFO = 1<<(PACF_MAX+2),
  PRIF_RESIZE_OFFLINE = 1<<(PACF_MAX+3)
} PRL_RESIZE_IMAGE_FLAGS;
```

File

PrlCommandsFlags.h

PRL_RUNNING_TASK_TYPE_ENUM Enumeration

Running task type enumeration.

Syntax

```
typedef enum _PRL_RUNNING_TASK_TYPE_ENUM {
  RTT_RUNNING_TASK_UNKNOWN = 0,
  RTT_RUNNING_TASK_CREATE_VM = 1,
  RTT_RUNNING_TASK_REGISTER_VM = 2,
  RTT_RUNNING_TASK_CLONE_VM = 3,
  RTT_RUNNING_TASK_DELETE_VM = 4,
  RTT_RUNNING_TASK_UNREG_VM = 5,
  RTT_RUNNING_TASK_CREATE_IMAGE = 6,
  RTT_RUNNING_TASK_RESTORE_VM = 7,
  RTT_RUNNING_TASK_MIGRATE_VM = 8,
  RTT_RUNNING_TASK_COPY_IMAGE = 9
} PRL_RUNNING_TASK_TYPE_ENUM;
```

File

PrlEnums.h

PRL_SCALE_VIEW_MODE Enumeration

Aspect ratio mode

Syntax

```
typedef enum _PRL_SCALE_VIEW_MODE {
  PSVM_OFF = 0,
  PSVM_AUTO = 1,
  PSVM_KEEP_ASPECT_RATIO = 2,
  PSVM_KEEP_ASPECT_RATIO_BY_EXPANDING = 3,
  PSVM_IGNORE_ASPECT_RATIO = 4
} PRL_SCALE_VIEW_MODE;
```

File

PrlEnums.h

PRL_SECURITY_LEVEL Enumeration

Security level enumeration.

Syntax

```
typedef enum _PRL_SECURITY_LEVEL {
  PSL_LOW_SECURITY = 0,
  PSL_NORMAL_SECURITY = 1<<0,
  PSL_HIGH_SECURITY = 1<<1
} PRL_SECURITY_LEVEL;
```

File

PrlCommandsFlags.h

Type Parameters

PSL_LOW_SECURITY

Plain TCP/IP (no encryption).

PSL_NORMAL_SECURITY

Most important data is sent and received using SSL over TCP/IP (user credentials during login, guest OS clipboard, etc.) Other data is sent and receive using plain TCP/IP with no encryption.

PSL_HIGH_SECURITY

All of the data is sent and received using SSL.

Remarks

Defines levels of security that can be used in client/server communications.

PRL_SERIAL_PORT_SOCKET_OPERATION_MODE Enumeration

Serial port operation mode when on socket

Syntax

```
typedef enum _PRL_SERIAL_PORT_SOCKET_OPERATION_MODE {
  PSP_SERIAL_SOCKET_SERVER = 0,
  PSP_SERIAL_SOCKET_CLIENT = 1
} PRL_SERIAL_PORT_SOCKET_OPERATION_MODE;
```

File

PrlEnums.h

PRL_SERVICE_STATUS_ENUM Enumeration

Virtuozzo Service statuses

Syntax

```
typedef enum _PRL_SERVICE_STATUS_ENUM {
  PSS_STARTED = 0,
  PSS_STOPPED = 1,
  PSS_NOT_INSTALLED = 2,
  PSS_UNKNOWN
} PRL_SERVICE_STATUS_ENUM;
```

File

PrlEnums.h

PRL_SET_USERPASSWD_FLAGS Enumeration

PrlVm_SetUserPasswd (p. 490) flags

Syntax

```
typedef enum _PRL_SET_USERPASSWD_FLAGS {
  PSPF_PASSWD_CRYPTED = 1<<(PACF_MAX+1)
} PRL_SET_USERPASSWD_FLAGS;
```

File

PrlCommandsFlags.h

Members

PSPF_PASSWD_CRYPTED
passwords are encrypted

PRL_SNAPSHOT_ELEMENT Structure

Virtuozzo snapshot tree element

Syntax

```
typedef struct __PRL_SNAPSHOT_ELEMENT {
  PRL_GUID m_GUID;
  struct __PRL_SNAPSHOT_ELEMENT* m_pNext;
  struct __PRL_SNAPSHOT_ELEMENT* m_pChild;
} PRL_SNAPSHOT_ELEMENT;
```

File

PrlDisk.h

PRL_SOUND_CHANNEL_TYPE Enumeration

Sound channel type

Syntax

```
typedef enum _PRL_SOUND_CHANNEL_TYPE {
  PSCT_AUTO = 0,
```

```
  PSCT_STEREO = 1,
  PSCT_SURROUND_5_1 = 2
} PRL_SOUND_CHANNEL_TYPE;
```

File

PrlEnums.h

PRL_SRV_SHUTDOWN_FLAGS Enumeration

PrlSrv_ShutdownEx (p. 298) flags

Syntax

```
typedef enum _PRL_SRV_SHUTDOWN_FLAGS {
  PSHF_FORCE_SHUTDOWN = 1<<(PACF_MAX+1),
  PSHF_DONT_WAIT_TO_COMPLETE = 1<<(PACF_MAX+2),
  PSHF_SUSPEND_VM_TO_PRAM = 1<<(PACF_MAX+3)
} PRL_SRV_SHUTDOWN_FLAGS;
```

File

PrlCommandsFlags.h

Members

PSHF_FORCE_SHUTDOWN

Allow shutdown with active users sessions and running vms

PSHF_DONT_WAIT_TO_COMPLETE

Send response immediately when shutdown started

PSHF_SUSPEND_VM_TO_PRAM

Use PRAM to suspend/resume VM if available

PRL_SWITCH_SNAPSHOT_FLAGS Enumeration

PrlVm_SwitchSnapshotEx flags

Syntax

```
typedef enum _PRL_SWITCH_SNAPSHOT_FLAGS {
  PSSF_SKIP_RESUME = 1<<(PACF_MAX+1)
} PRL_SWITCH_SNAPSHOT_FLAGS;
```

File

PrlCommandsFlags.h

Members

PSSF_SKIP_RESUME

skip resume stage for Vm in the

- running/paused state in the snapshot

PRL_TASKBAR_STATES Enumeration

Taskbar states

Syntax

```
typedef enum _PRL_TASKBAR_STATES {
  PTBS_UNKNOWN = -1,
  PTBS_HIDE = 0,
  PTBS_SHOW = 1,
  PTBS_DEFAULT = 2
} PRL_TASKBAR_STATES;
```

File

PrlEnums.h

PRL_TIS_RECORD_FIELD Enumeration

Virtuozzo Tools Information Service record fields

Syntax

```
typedef enum _PRL_TIS_RECORD_FIELD {
  PTIS_RECORD_UID = 1<<0,
  PTIS_RECORD_NAME = 1<<1,
  PTIS_RECORD_INFO = 1<<2,
  PTIS_RECORD_TEXT = 1<<3,
  PTIS_RECORD_DATA = 1<<4,
  PTIS_RECORD_STATE = 1<<5,
  PTIS_RECORD_TIME = 1<<6,
  PTIS_RECORD_OWNER = 1<<7,
  PTIS_RECORD_FLAGS = 1<<8,
  PTIS_RECORD_REMOVED = 1<<30,
  PTIS_RECORD_CLEARED = 1<<31
} PRL_TIS_RECORD_FIELD;
```

File

PrlEnums.h

Members

PTIS_RECORD_REMOVED
: set when record removed

PTIS_RECORD_CLEARED
: set only with PTIS_RECORD_REMOVED bit when entire database was cleared

PRL_TIS_RECORD_STATE Enumeration

Virtuozzo Tools Information Service record states

Syntax

```
typedef enum _PRL_TIS_RECORD_STATE {
```

```
  PTIS_RECORD_EMPTY = 0,
  PTIS_RECORD_ACTIVE = 1,
  PTIS_RECORD_CANCELED = 2
} PRL_TIS_RECORD_STATE;
```

File

PrlEnums.h

PRL_TOOLS_SIA_OPEN_IN_TYPES Enumeration

Shared Internet Applications 'open in' type PSIA_OPEN_DEFAULT : Open host link with host app; Open guest link with guest app PSIA_OPEN_IN_GUEST : Open all links with guest apps; PSIA_OPEN_IN_HOST : Open all links with host apps;

Syntax

```
typedef enum _PRL_TOOLS_SIA_OPEN_IN_TYPES {
  PSIA_OPEN_DEFAULT = 0,
  PSIA_OPEN_IN_GUEST = 1,
  PSIA_OPEN_IN_HOST = 2
} PRL_TOOLS_SIA_OPEN_IN_TYPES;
```

File

PrlEnums.h

PRL_TOOLS_UTILITY_COMMAND_FLAG Enumeration

Virtuozzo Tools Utility command flags

Syntax

```
typedef enum _PRL_TOOLS_UTILITY_COMMAND_FLAG {
  PTU_FLG_ZERO = (1 << 0)
} PRL_TOOLS_UTILITY_COMMAND_FLAG;
```

File

PrlCommandsFlags.h

Members

PTU_FLG_ZERO

@brief Flag is useless now, but placed here to show how and where it must be

PRL_TOOLS_UTILITY_COMMAND_ID Enumeration

Virtuozzo Tools Utility commands identifiers

Syntax

```
typedef enum _PRL_TOOLS_UTILITY_COMMAND_ID {
  PTU_CMD_INVALID = 0,
  PTU_CMD_PTU_ALIVE = 1
} PRL_TOOLS_UTILITY_COMMAND_ID;
```

File

PrlEnums.h

Members

PTU_CMD_INVALID
brief Identifier of invalid command

PTU_CMD_PTU_ALIVE
@brief Can be sent to indicate that Virtuozzo Tools Utility subsystem is alive and ready to work

PRL_UNDO_DISKS_MODE Enumeration

VM undo disks mode: PUD_DISABLE_UNDO_DISKS : disbale undo disks PUD_REVERSE_CHANGES : at stopping VM revert all disk changes PUD_COMMIT_CHANGES : at stopping VM apply all disk changes PUD_PROMPT_BEHAVIOUR : ask user which mode reverse or commit will be applied

Syntax

```
typedef enum _PRL_UNDO_DISKS_MODE {
  PUD_DISABLE_UNDO_DISKS = 0,
  PUD_REVERSE_CHANGES = 1,
  PUD_COMMIT_CHANGES = 2,
  PUD_PROMPT_BEHAVIOUR = 3
} PRL_UNDO_DISKS_MODE;
```

File

PrlEnums.h

PRL_UPDATE_LICENSE_FLAGS Enumeration

Update/Install license operations: Uses for PrlSrv_UpdateLicenseEx (p. 310) operation

Syntax

```
typedef enum _PRL_UPDATE_LICENSE_FLAGS {
  PUPLF_KICK_TO_UDPATE_CURR_FILE_LICENSE = 1<<(PACF_MAX+1),
  PUPLF_ACTIVATE_PERMANENT_LICENSE_ONLINE = 1<<(PACF_MAX+2),
  PUPLF_ACTIVATE_PERMANENT_LICENSE_OFFLINE = 1<<(PACF_MAX+3),
  PUPLF_CHECK_KEY_ONLY = 1<<(PACF_MAX+4),
  PUPLF_DEACTIVATE_CURRENT_LICENSE = 1<<(PACF_MAX+5),
  PUPLF_ACTIVATE_PRECACHED_KEY = 1<<(PACF_MAX+6),
  PUPLF_STORE_PRECACHED_KEY = 1<<(PACF_MAX+7),
  PUPLF_REMOVE_PRECACHED_KEY = 1<<(PACF_MAX+8),
  PUPLF_DEACTIVATE_CURRENT_LICENSE_OFFLINE_ALLOWED = 1<<(PACF_MAX+9),
  PUPLF_CHECK_KEY_ONLY_VOLUME = 1<<(PACF_MAX+10),
  PUPLF_REGISTER_CURRENT_LICENSE = 1<<(PACF_MAX+11)
} PRL_UPDATE_LICENSE_FLAGS;
```

File

PrlCommandsFlags.h

PRL_USB_DEVICE_AUTO_CONNECT_OPTION Enumeration

USB autoconnect devices options

Syntax

```
typedef enum _PRL_USB_DEVICE_AUTO_CONNECT_OPTION {
  PUD_CONNECT_TO_PRIMARY_OS = 0,
  PUD_CONNECT_TO_GUEST_OS = 1,
  PUD_ASK_USER_WHAT_TODO = 2
} PRL_USB_DEVICE_AUTO_CONNECT_OPTION;
```

File

PrlEnums.h

PRL_USB_DEVICE_CONNECT_REASON Enumeration

USB device connect/disconnect reason

Syntax

```
typedef enum _PRL_USB_DEVICE_CONNECT_REASON {
  PUD_CONNECTED_MANUALLY = 0,
  PUD_CONNECTED_AUTOMATICALLY = 1
} PRL_USB_DEVICE_CONNECT_REASON;
```

File

PrlEnums.h

PRL_USB_DEVICE_TYPE Enumeration

USB device type

Syntax

```
typedef enum _PRL_USB_DEVICE_TYPE {
  PUDT_OTHER = 0,
  PUDT_HUB = 1,
  PUDT_VIDEO = 2,
  PUDT_FOTO = 3,
  PUDT_AUDIO = 4,
  PUDT_PRINTER = 5,
  PUDT_SCANNER = 6,
  PUDT_BLUETOOTH = 7,
  PUDT_WIRELESS = 8,
  PUDT_COMMUNICATION = 9,
  PUDT_KEYBOARD = 10,
  PUDT_MOUSE = 11,
  PUDT_SMART_CARD = 12,
  PUDT_DISK_STORAGE = 13,
  PUDT_ATAPI_STORAGE = 14,
  PUDT_APPLE_IPHONE = 1000,
  PUDT_APPLE_IPOD = 1001,
  PUDT_APPLE_IPAD = 1002,
```

```
    PUDT_APPLE_IETH = 1003,
    PUDT_RIM_BLACKBERRY = 2000,
    PUDT_GARMIN_GPS = 3000
} PRL_USB_DEVICE_TYPE;
```

File

PrlEnums.h

Members

PUDT_OTHER
unknown device type

PUDT_HUB
usb hub

PUDT_VIDEO
streaming usb video device

PUDT_FOTO
still image usb video device

PUDT_AUDIO
usb audio device

PUDT_PRINTER
usb printer

PUDT_SCANNER
usb scaner

PUDT_BLUETOOTH
usb bluetooth

PUDT_WIRELESS
usb wireless device

PUDT_COMMUNICATION
other usb communication device

PUDT_KEYBOARD
usb keyboard

PUDT_MOUSE
usb mouse

PUDT_SMART_CARD
usb smartcard reader

PUDT_DISK_STORAGE
usb storage

PUDT_ATAPI_STORAGE
usb CD/DVD-ROMs Some vendor specific usb device classes

PUDT_APPLE_IPHONE
apple iphone

PUDT_APPLE_IPOD
apple ipod

PUDT_APPLE_IPAD
apple ipad

PUDT_APPLE_IETH

apple ieth

PUDT_RIM_BLACKBERRY

rim blackberry

PUDT_GARMIN_GPS

garmin gps

PRL_VCMMD_COMMAND_FLAGS Enumeration

Syntax

```
typedef enum _PRL_VCMMD_COMMAND_FLAGS {
  VCMMD_COMMAND_RUNTIME = 1<<(PACF_MAX+1)
} PRL_VCMMD_COMMAND_FLAGS;
```

File

PrlCommandsFlags.h

PRL_VERSION_ENUM Enumeration

Virtuozzo license versions enumeration.

Syntax

```
typedef enum _PRL_VERSION_ENUM {
  PRL_VERSION_ANY = 0,
  PRL_VERSION_3X = 1,
  PRL_VERSION_4X = 2,
  PRL_VERSION_5X = 3,
  PRL_VERSION_6X = 4,
  PRL_VERSION_7X = 5,
  PRL_VERSION_8X = 6,
  PRL_VERSION_9X = 7,
  PRL_VERSION_10X = 8
} PRL_VERSION_ENUM;
```

File

PrlEnums.h

PRL_VIDEO_3D_ACCELERATION Enumeration

Video 3D acceleration

Syntax

```
typedef enum _PRL_VIDEO_3D_ACCELERATION {
  P3D_DISABLED = 0,
  P3D_ENABLED_HIGHEST = 1,
  P3D_ENABLED_DX9 = 2
} PRL_VIDEO_3D_ACCELERATION;
```

File

PrlEnums.h

PRL_VIDEO_RECEIVER_CONNECTION_STATE Enumeration

Streaming connection status enum

Syntax

```
enum PRL_VIDEO_RECEIVER_CONNECTION_STATE {
  PVRCS_CONNECTED = 0,
  PVRCS_DISCONNECTED = 1,
  PVRCS_NO_CONNECTION = 2,
  PVRCS_CONNECTING = 3
};
```

File

PrlEnums.h

PRL_VIRTUAL_MACHINE_PROFILE Enumeration

Virtual machine profiles enum

Syntax

```
typedef enum _PRL_VIRTUAL_MACHINE_PROFILE {
  VMP_PRODUCTIVITY = 0,
  VMP_GAMES = 1,
  VMP_SOFTWARE_DEVELOPER = 2,
  VMP_DESIGNER = 3
} PRL_VIRTUAL_MACHINE_PROFILE;
```

File

PrlEnums.h

PRL_VIRTUAL_NETWORK_FLAGS Enumeration

PrlSrv_AddVirtualNetwork (p. 224) flangs

Syntax

```
typedef enum _PRL_VIRTUAL_NETWORK_FLAGS {
  PRL_USE_VNET_NAME_FOR_BRIDGE_NAME = 1<<(PACF_MAX+1)
} PRL_VIRTUAL_NETWORK_FLAGS;
```

File

PrlCommandsFlags.h

PRL_VMCTG_EVENT Enumeration

Tools events, passed to the callback (see PRL_VMCTG_TOOL_DESC and PRL_TOOL_HANDLER_PTR above)

Syntax

```
typedef enum _PRL_VMCTG_EVENT {
  TEV_DATA_RECEIVED,
  TEV_CONNECTED,
  TEV_DISCONNECTED
} PRL_VMCTG_EVENT;
```

File

PrlEnums.h

Members

TEV_DATA_RECEIVED

pData is PRL_TOOL_DATA_RECEIVED

TEV_CONNECTED

pData is null

TEV_DISCONNECTED

pData is null

PRL_VTD_STATUS_CODE Enumeration

VTD status code

Syntax

```
typedef enum _PRL_VTD_STATUS_CODE {
  PVSC_DMAR_NOT_FOUND = -5,
  PVSC_DMAR_ERROR = -4,
  PVSC_ACPI_ERROR = -3,
  PVSC_HARDWARE_ERROR = -2,
  PVSC_INTERNAL_ERROR = -1,
  PVSC_NOT_PRESENT = 0,
  PVSC_ENABLED = 1
} PRL_VTD_STATUS_CODE;
```

File

PrlEnums.h

PRL_WINDOW_CORNERS Enumeration

Window corners

Syntax

```
typedef enum _PRL_WINDOW_CORNERS {
  PWC_TOP_LEFT_CORNER = 0,
  PWC_TOP_RIGHT_CORNER = 1,
```

```
  PWC_BOTTOM_LEFT_CORNER = 2,
  PWC_BOTTOM_RIGHT_CORNER = 3
} PRL_WINDOW_CORNERS;
```

File

PrlEnums.h

PRL_WINDOW_EDGE_MASK Enumeration

Edges where taskbar can be placed (in coherence mode).

Syntax

```
typedef enum _PRL_WINDOW_EDGE_MASK {
  WINDOW_LEFT_EDGE_MASK = (0x1L << 0),
  WINDOW_TOP_EDGE_MASK = (0x1L << 1),
  WINDOW_RIGHT_EDGE_MASK = (0x1L << 2),
  WINDOW_BOTTOM_EDGE_MASK = (0x1L << 3),
  WINDOW_ALL_EDGE_MASK =
WINDOW_LEFT_EDGE_MASK|WINDOW_TOP_EDGE_MASK|WINDOW_RIGHT_EDGE_MASK|WINDOW_BOTTOM_EDGE_MASK
} PRL_WINDOW_EDGE_MASK;
```

File

PrlEnums.h

Other Structs

Enumerations

	Name	Description
	_VIRTUAL_MACHINE_ADDITION_STATE (p. 1103)	Virtual machine addition states enumeration, may be combined with bitwise
	_VIRTUAL_MACHINE_STATE (p. 1103)	Virtual machine states enumeration
	VIRTUAL_MACHINE_ADDITION_STATE (p. 1104)	Virtual machine addition states enumeration, may be combined with bitwise
	VIRTUAL_MACHINE_STATE (p. 1104)	Virtual machine states enumeration

Structures

Name	Description
IPluginInfo (p. 1104)	Generic information about plugin

_VIRTUAL_MACHINE_ADDITION_STATE Enumeration

Virtual machine addition states enumeration, may be combined with bitwise

Syntax

```
enum _VIRTUAL_MACHINE_ADDITION_STATE {
  VMAS_NOSTATE = 1<<0,
```

```
  VMAS_BACKUPING = 1<<1,
  VMAS_RESTORING_FROM_BACKUP = 1<<2,
  VMAS_CLONING = 1<<5,
  VMAS_MOVING = 1<<6,
  VMAS_MAX = 1<<31
};
```

File

PrlEnums.h

_VIRTUAL_MACHINE_STATE Enumeration

Virtual machine states enumeration

Syntax

```
enum _VIRTUAL_MACHINE_STATE {
  VMS_UNKNOWN = 0,
  VMS_STOPPED = 0x30000001,
  VMS_STARTING = 0x30000002,
  VMS_RESTORING = 0x30000003,
  VMS_RUNNING = 0x30000004,
  VMS_PAUSED = 0x30000005,
  VMS_SUSPENDING = 0x30000006,
  VMS_STOPPING = 0x30000007,
  VMS_COMPACTING = 0x30000008,
  VMS_SUSPENDED = 0x30000009,
  VMS_SNAPSHOTING = 0x3000000A,
  VMS_RESETTING = 0x3000000B,
  VMS_PAUSING = 0x3000000C,
  VMS_CONTINUING = 0x3000000D,
  VMS_MIGRATING = 0x3000000E,
  VMS_DELETING_STATE = 0x3000000F,
  VMS_RESUMING = 0x30000010,
  VMS_SUSPENDING_SYNC = 0x30000011,
  VMS_RECONNECTING = 0x30000012,
  VMS_MOUNTED = 0x30000013
};
```

File

PrlEnums.h

IPluginInfo Structure

Generic information about plugin

Syntax

```
typedef struct _IPluginInfo {
  PRL_CONST_STR Vendor;
  PRL_CONST_STR Copyright;
  PRL_CONST_STR DescShort;
  PRL_CONST_STR DescLong;
  PRL_UINT32 Major;
  PRL_UINT32 Minor;
  PRL_UINT32 Build;
  PRL_GUID Type;
} IPluginInfo;
```

File

PrlPluginClasses.h

VIRTUAL_MACHINE_ADDITION_STATE Enumeration

Virtual machine addition states enumeration, may be combined with bitwise

Syntax

```
typedef enum _VIRTUAL_MACHINE_ADDITION_STATE {
  VMAS_NOSTATE = 1<<0,
  VMAS_BACKUPING = 1<<1,
  VMAS_RESTORING_FROM_BACKUP = 1<<2,
  VMAS_CLONING = 1<<5,
  VMAS_MOVING = 1<<6,
  VMAS_MAX = 1<<31
} VIRTUAL_MACHINE_ADDITION_STATE;
```

File

PrlEnums.h

VIRTUAL_MACHINE_STATE Enumeration

Virtual machine states enumeration

Syntax

```
typedef enum _VIRTUAL_MACHINE_STATE {
  VMS_UNKNOWN = 0,
  VMS_STOPPED = 0x30000001,
  VMS_STARTING = 0x30000002,
  VMS_RESTORING = 0x30000003,
  VMS_RUNNING = 0x30000004,
  VMS_PAUSED = 0x30000005,
  VMS_SUSPENDING = 0x30000006,
  VMS_STOPPING = 0x30000007,
  VMS_COMPACTING = 0x30000008,
  VMS_SUSPENDED = 0x30000009,
  VMS_SNAPSHOTING = 0x3000000A,
  VMS_RESETTING = 0x3000000B,
  VMS_PAUSING = 0x3000000C,
  VMS_CONTINUING = 0x3000000D,
  VMS_MIGRATING = 0x3000000E,
  VMS_DELETING_STATE = 0x3000000F,
  VMS_RESUMING = 0x30000010,
  VMS_SUSPENDING_SYNC = 0x30000011,
  VMS_RECONNECTING = 0x30000012,
  VMS_MOUNTED = 0x30000013
} VIRTUAL_MACHINE_STATE;
```

File

PrlEnums.h

Macros

Topics

Name	Description
PET Macros (p. 1106)	
PRL_DISK Macros (p. 1128)	
PRL_ERR Macros (p. 1137)	
PRL_GUEST Macros (p. 1499)	
PRL_IMAGE Macros (p. 1503)	
PRL_NET Macros (p. 1508)	
PRL_QUEST Macros (p. 1521)	
PRL_QUESTION Macros (p. 1523)	
PRL_SCAN_CODE Macros (p. 1526)	
PRL_STATE Macros (p. 1561)	
PRL_WARN Macros (p. 1564)	
PRL_WRN Macros (p. 1575)	
PVS_GUEST Macros (p. 1576)	
Other PRL Macros (p. 1620)	
Other Macros (p. 1654)	

PET Macros

Macros

Name	Description
PET_ANSWER_APPEND (p. 1108)	
PET_ANSWER_BREAK (p. 1109)	
PET_ANSWER_BROWSE (p. 1109)	
PET_ANSWER_CANCEL (p. 1109)	
PET_ANSWER_CHANGE (p. 1109)	
PET_ANSWER_COMMIT (p. 1110)	
PET_ANSWER_COMPACT (p. 1110)	
PET_ANSWER_CONTINUE (p. 1110)	
PET_ANSWER_COPIED (p. 1110)	
PET_ANSWER_CREATE (p. 1110)	
PET_ANSWER_CREATE_NEW (p. 1111)	
PET_ANSWER_DISCONNECT_ANYWAY (p. 1111)	
PET_ANSWER_DONT_CHANGE (p. 1111)	
PET_ANSWER_LATER (p. 1111)	
PET_ANSWER_MOVED (p. 1111)	

PET_ANSWER_NO (p. 1112)	
PET_ANSWER_OK (p. 1112)	• Answers +++ ALIGNED VALUE TO 16000
PET_ANSWER_OVERIDE (p. 1112)	
PET_ANSWER_REPLACE (p. 1112)	
PET_ANSWER_RESTART (p. 1112)	
PET_ANSWER_RESTART_NOW (p. 1113)	
PET_ANSWER_RESTORE (p. 1113)	
PET_ANSWER_RESUME (p. 1113)	
PET_ANSWER_RETRY (p. 1113)	
PET_ANSWER_REVERT (p. 1114)	
PET_ANSWER_SHUTDOWN (p. 1114)	
PET_ANSWER_SKIP (p. 1114)	
PET_ANSWER_STARTVM (p. 1114)	
PET_ANSWER_STOP (p. 1114)	
PET_ANSWER_STOP_AND_RESTORE (p. 1115)	
PET_ANSWER_SUSPEND (p. 1115)	
PET_ANSWER_SUSPEND_ANYWAY (p. 1115)	
PET_ANSWER_USE_CURRENT (p. 1115)	
PET_ANSWER_YES (p. 1115)	
PET_ERR_WARN_REACH_OVERCOMMIT_STATE_ON_SETMEM (p. 1116)	
PET_QUESTION_ALLOW_TO_SUSPEND_VM_WITH_BOOTCAMP_DISK (p. 1116)	
PET_QUESTION_APPLIANCE_CORRUPTED_INSTALLATION (p. 1116)	
PET_QUESTION_APPLY_WHEN_HYP_CANNOT_ALLOC_VMS_MEM (p. 1116)	
PET_QUESTION_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE (p. 1117)	
PET_QUESTION_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE (p. 1117)	
PET_QUESTION_BOOTCAMP_HELPER_INIT_FAILURE (p. 1117)	
PET_QUESTION_BOOTCAMP_HELPER_OS_UNSUPPORTED (p. 1118)	
PET_QUESTION_CANCEL_CLONE_OPERATION (p. 1118)	
PET_QUESTION_CANCEL_CLONE_TO_TEMPLATE_OPERATION (p. 1118)	
PET_QUESTION_CANCEL_CONVERT_3RD_VM_OPERATION (p. 1118)	
PET_QUESTION_CANCEL_DEPLOY_OPERATION (p. 1119)	
PET_QUESTION_CANCEL_IMPORT_BOOTCAMP_OPERATION (p. 1119)	
PET_QUESTION_CHANGE_VM_MIGRATION_TYPE (p. 1119)	
PET_QUESTION_COMMON_HDD_ERROR (p. 1119)	
PET_QUESTION_COMPACT_VM_DISKS (p. 1120)	
PET_QUESTION_CONTINUE_IF_HVT_DISABLED (p. 1120)	
PET_QUESTION_CREATE_NEW_MAC_ADDRESS (p. 1120)	
PET_QUESTION_CREATE_OS2_GUEST_WITHOUT_FDD_IMAGE (p. 1120)	
PET_QUESTION_CREATE_VM_FROM_LION_RECOVERY_PART (p. 1121)	

PET_QUESTION_DELAYED_COMPACT_VM_DISKS (p. 1121)	
PET_QUESTION_DELETE_FILES_OUT_OF_VM_DIR (p. 1121)	
PET_QUESTION_DELETE_PARALLELS_FILES_IN_VM_DIR (p. 1121)	
PET_QUESTION_DELETE_VM_WITH_CORRUPTED_CONFIG (p. 1122)	
PET_QUESTION_DO_YOU_WANT_TO_OVERWRITE_FILE (p. 1122)	
PET_QUESTION_FORCE_COMPACT_VM_DISKS (p. 1122)	
PET_QUESTION_FREE_SIZE_FOR_COMPRESSED_DISK (p. 1122)	
PET_QUESTION_MAC_PHYSICAL_MEMORY_MAP_BUG (p. 1123)	
PET_QUESTION_OLD_CONFIG_CONVERTION (p. 1123)	
PET_QUESTION_ON_QUERY_END_SESSION (p. 1123)	
PET_QUESTION_ON_QUERY_END_SESSION_RESTRICTED (p. 1123)	
PET_QUESTION_REACH_OVERCOMMIT_STATE (p. 1124)	
PET_QUESTION_REACH_OVERCOMMIT_STATE_ON_SETMEM (p. 1124)	
PET_QUESTION_REBOOT_HOST_ON_PCI_DRIVER_INSTALL_OR_REVERT (p. 1124)	
PET_QUESTION_REGISTER_USED_VM (p. 1124)	
PET_QUESTION_REGISTER_VM_TEMPLATE (p. 1125)	
PET_QUESTION_RESTART_VM_GUEST_TO_COMPACT (p. 1125)	
PET_QUESTION_RESTORE_VM_CONFIG_FROM_BACKUP (p. 1125)	
PET_QUESTION_SAMPLE_1 (p. 1125)	• Questions +++ ALIGNED VALUE TO 13000
PET_QUESTION_SAMPLE_2 (p. 1126)	
PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE (p. 1126)	
PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE_CPU (p. 1126)	
PET_QUESTION_STOP_VM_TO_COMPACT (p. 1126)	
PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE (p. 1126)	
PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE_CPU (p. 1127)	
PET_QUESTION_SWITCH_OFF_AUTO_COMPRESS (p. 1127)	
PET_QUESTION_TXT_ERROR (p. 1127)	
PET_QUESTION_UNDO_DISKS_MODE (p. 1127)	
PET_QUESTION_VM_COPY_OR_MOVE (p. 1128)	
PET_QUESTION_VM_REBOOT_REQUIRED_BY_PRL_TOOLS (p. 1128)	
PET_QUESTION_VM_ROOT_DIRECTORY_NOT_EXISTS (p. 1128)	

PET_ANSWER_APPEND Macro

Syntax

```
#define PET_ANSWER_APPEND PRL_RESULT_DECLARE_SUCCESS(16011)
```

File

PrlErrorsValues.h

PET_ANSWER_BREAK Macro

Syntax

```
#define PET_ANSWER_BREAK PRL_RESULT_DECLARE_SUCCESS(16004)
```

File

PrlErrorsValues.h

PET_ANSWER_BROWSE Macro

Syntax

```
#define PET_ANSWER_BROWSE PRL_RESULT_DECLARE_SUCCESS(16028)
```

File

PrlErrorsValues.h

PET_ANSWER_CANCEL Macro

Syntax

```
#define PET_ANSWER_CANCEL PRL_RESULT_DECLARE_SUCCESS(16001)
```

File

PrlErrorsValues.h

PET_ANSWER_CHANGE Macro

Syntax

```
#define PET_ANSWER_CHANGE PRL_RESULT_DECLARE_SUCCESS(16032)
```

File

PrlErrorsValues.h

PET_ANSWER_COMMIT Macro

Syntax

```
#define PET_ANSWER_COMMIT PRL_RESULT_DECLARE_SUCCESS(16016)
```

File

PrlErrorsValues.h

PET_ANSWER_COMPACT Macro

Syntax

```
#define PET_ANSWER_COMPACT PRL_RESULT_DECLARE_SUCCESS(16024)
```

File

PrlErrorsValues.h

PET_ANSWER_CONTINUE Macro

Syntax

```
#define PET_ANSWER_CONTINUE PRL_RESULT_DECLARE_SUCCESS(16008)
```

File

PrlErrorsValues.h

PET_ANSWER_COPIED Macro

Syntax

```
#define PET_ANSWER_COPIED PRL_RESULT_DECLARE_SUCCESS(16013)
```

File

PrlErrorsValues.h

PET_ANSWER_CREATE Macro

Syntax

```
#define PET_ANSWER_CREATE PRL_RESULT_DECLARE_SUCCESS(16019)
```

File

PrlErrorsValues.h

PET_ANSWER_CREATE_NEW Macro

Syntax

```
#define PET_ANSWER_CREATE_NEW PRL_RESULT_DECLARE_SUCCESS(16009)
```

File

PrlErrorsValues.h

PET_ANSWER_DISCONNECT_ANYWAY Macro

Syntax

```
#define PET_ANSWER_DISCONNECT_ANYWAY PRL_RESULT_DECLARE_SUCCESS(16018)
```

File

PrlErrorsValues.h

PET_ANSWER_DONT_CHANGE Macro

Syntax

```
#define PET_ANSWER_DONT_CHANGE PRL_RESULT_DECLARE_SUCCESS(16033)
```

File

PrlErrorsValues.h

PET_ANSWER_LATER Macro

Syntax

```
#define PET_ANSWER_LATER PRL_RESULT_DECLARE_SUCCESS(16023)
```

File

PrlErrorsValues.h

PET_ANSWER_MOVED Macro

Syntax

```
#define PET_ANSWER_MOVED PRL_RESULT_DECLARE_SUCCESS(16014)
```

File

PrlErrorsValues.h

PET_ANSWER_NO Macro

Syntax

```
#define PET_ANSWER_NO PRL_RESULT_DECLARE_SUCCESS(16003)
```

File

PrlErrorsValues.h

PET_ANSWER_OK Macro

- Answers

+++ ALIGNED VALUE TO 16000

Syntax

```
#define PET_ANSWER_OK PRL_RESULT_DECLARE_SUCCESS(16000)
```

File

PrlErrorsValues.h

PET_ANSWER_OVERIDE Macro

Syntax

```
#define PET_ANSWER_OVERIDE PRL_RESULT_DECLARE_SUCCESS(16005)
```

File

PrlErrorsValues.h

PET_ANSWER_REPLACE Macro

Syntax

```
#define PET_ANSWER_REPLACE PRL_RESULT_DECLARE_SUCCESS(16012)
```

File

PrlErrorsValues.h

PET_ANSWER_RESTART Macro

Syntax

```
#define PET_ANSWER_RESTART PRL_RESULT_DECLARE_SUCCESS(16030)
```

File

PrlErrorsValues.h

PET_ANSWER_RESTART_NOW Macro

Syntax

```
#define PET_ANSWER_RESTART_NOW PRL_RESULT_DECLARE_SUCCESS(16022)
```

File

PrlErrorsValues.h

PET_ANSWER_RESTORE Macro

Syntax

```
#define PET_ANSWER_RESTORE PRL_RESULT_DECLARE_SUCCESS(16027)
```

File

PrlErrorsValues.h

PET_ANSWER_RESUME Macro

Syntax

```
#define PET_ANSWER_RESUME PRL_RESULT_DECLARE_SUCCESS(16031)
```

File

PrlErrorsValues.h

PET_ANSWER_RETRY Macro

Syntax

```
#define PET_ANSWER_RETRY PRL_RESULT_DECLARE_SUCCESS(16029)
```

File

PrlErrorsValues.h

PET_ANSWER_REVERT Macro

Syntax

```
#define PET_ANSWER_REVERT PRL_RESULT_DECLARE_SUCCESS(16017)
```

File

PrlErrorsValues.h

PET_ANSWER_SHUTDOWN Macro

Syntax

```
#define PET_ANSWER_SHUTDOWN PRL_RESULT_DECLARE_SUCCESS(16006)
```

File

PrlErrorsValues.h

PET_ANSWER_SKIP Macro

Syntax

```
#define PET_ANSWER_SKIP PRL_RESULT_DECLARE_SUCCESS(16020)
```

File

PrlErrorsValues.h

PET_ANSWER_STARTVM Macro

Syntax

```
#define PET_ANSWER_STARTVM PRL_RESULT_DECLARE_SUCCESS(16015)
```

File

PrlErrorsValues.h

PET_ANSWER_STOP Macro

Syntax

```
#define PET_ANSWER_STOP PRL_RESULT_DECLARE_SUCCESS(16007)
```

File

PrlErrorsValues.h

PET_ANSWER_STOP_AND_RESTORE Macro

Syntax

```
#define PET_ANSWER_STOP_AND_RESTORE PRL_RESULT_DECLARE_SUCCESS(16026)
```

File

PrlErrorsValues.h

PET_ANSWER_SUSPEND Macro

Syntax

```
#define PET_ANSWER_SUSPEND PRL_RESULT_DECLARE_SUCCESS(16025)
```

File

PrlErrorsValues.h

PET_ANSWER_SUSPEND_ANYWAY Macro

Syntax

```
#define PET_ANSWER_SUSPEND_ANYWAY PRL_RESULT_DECLARE_SUCCESS(16021)
```

File

PrlErrorsValues.h

PET_ANSWER_USE_CURRENT Macro

Syntax

```
#define PET_ANSWER_USE_CURRENT PRL_RESULT_DECLARE_SUCCESS(16010)
```

File

PrlErrorsValues.h

PET_ANSWER_YES Macro

Syntax

```
#define PET_ANSWER_YES PRL_RESULT_DECLARE_SUCCESS(16002)
```

File

PrlErrorsValues.h

PET_ERR_WARN_REACH_OVERCOMMIT_STATE_ON_SETMEM
Macro

Syntax

```
#define PET_ERR_WARN_REACH_OVERCOMMIT_STATE_ON_SETMEM PRL_RESULT_DECLARE_SUCCESS(441)
```

File

PrlErrorsValues.h

PET_QUESTION_ALLOW_TO_SUSPEND_VM_WITH_BOOTCAMP_DISK
Macro

Syntax

```
#define PET_QUESTION_ALLOW_TO_SUSPEND_VM_WITH_BOOTCAMP_DISK PRL_RESULT_DECLARE_SUCCESS(13022)
```

File

PrlErrorsValues.h

PET_QUESTION_APPLIANCE_CORRUPTED_INSTALLATION
Macro

Syntax

```
#define PET_QUESTION_APPLIANCE_CORRUPTED_INSTALLATION PRL_RESULT_DECLARE_SUCCESS(13037)
```

File

PrlErrorsValues.h

PET_QUESTION_APPLY_WHEN_HYP_CANNOT_ALLOC_VMS_MEM
Macro

Syntax

```
#define PET_QUESTION_APPLY_WHEN_HYP_CANNOT_ALLOC_VMS_MEM PRL_RESULT_DECLARE_SUCCESS(13021)
```

File

PrlErrorsValues.h

PET_QUESTION_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE
Macro

Syntax

```
#define PET_QUESTION_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE PRL_RESULT_DECLARE_ERROR(13036)
```

File

PrlErrorsValues.h

PET_QUESTION_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE
Macro

Syntax

```
#define PET_QUESTION_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE PRL_RESULT_DECLARE_ERROR(13035)
```

File

PrlErrorsValues.h

PET_QUESTION_BOOTCAMP_HELPER_INIT_FAILURE
Macro

Syntax

```
#define PET_QUESTION_BOOTCAMP_HELPER_INIT_FAILURE PRL_RESULT_DECLARE_SUCCESS(13015)
```

File

PrlErrorsValues.h

PET_QUESTION_BOOTCAMP_HELPER_OS_UNSUPPORTED Macro

Syntax

```
#define PET_QUESTION_BOOTCAMP_HELPER_OS_UNSUPPORTED PRL_RESULT_DECLARE_SUCCESS(13016)
```

File

PrlErrorsValues.h

PET_QUESTION_CANCEL_CLONE_OPERATION Macro

Syntax

```
#define PET_QUESTION_CANCEL_CLONE_OPERATION PRL_RESULT_DECLARE_SUCCESS(13024)
```

File

PrlErrorsValues.h

PET_QUESTION_CANCEL_CLONE_TO_TEMPLATE_OPERATION Macro

Syntax

```
#define PET_QUESTION_CANCEL_CLONE_TO_TEMPLATE_OPERATION PRL_RESULT_DECLARE_SUCCESS(13025)
```

File

PrlErrorsValues.h

PET_QUESTION_CANCEL_CONVERT_3RD_VM_OPERATION Macro

Syntax

```
#define PET_QUESTION_CANCEL_CONVERT_3RD_VM_OPERATION PRL_RESULT_DECLARE_SUCCESS(13046)
```

File

PrlErrorsValues.h

PET_QUESTION_CANCEL_DEPLOY_OPERATION Macro

Syntax

```
#define PET_QUESTION_CANCEL_DEPLOY_OPERATION PRL_RESULT_DECLARE_SUCCESS(13026)
```

File

PrlErrorsValues.h

PET_QUESTION_CANCEL_IMPORT_BOOTCAMP_OPERATION Macro

Syntax

```
#define PET_QUESTION_CANCEL_IMPORT_BOOTCAMP_OPERATION PRL_RESULT_DECLARE_SUCCESS(13044)
```

File

PrlErrorsValues.h

PET_QUESTION_CHANGE_VM_MIGRATION_TYPE Macro

Syntax

```
#define PET_QUESTION_CHANGE_VM_MIGRATION_TYPE PRL_RESULT_DECLARE_SUCCESS(13012)
```

File

PrlErrorsValues.h

PET_QUESTION_COMMON_HDD_ERROR Macro

Syntax

```
#define PET_QUESTION_COMMON_HDD_ERROR PRL_RESULT_DECLARE_SUCCESS(13047)
```

File

PrlErrorsValues.h

PET_QUESTION_COMPACT_VM_DISKS Macro

Syntax

```
#define PET_QUESTION_COMPACT_VM_DISKS PRL_RESULT_DECLARE_SUCCESS(13030)
```

File

PrlErrorsValues.h

PET_QUESTION_CONTINUE_IF_HVT_DISABLED Macro

Syntax

```
#define PET_QUESTION_CONTINUE_IF_HVT_DISABLED PRL_RESULT_DECLARE_SUCCESS(13031)
```

File

PrlErrorsValues.h

PET_QUESTION_CREATE_NEW_MAC_ADDRESS Macro

Syntax

```
#define PET_QUESTION_CREATE_NEW_MAC_ADDRESS PRL_RESULT_DECLARE_SUCCESS(13011)
```

File

PrlErrorsValues.h

PET_QUESTION_CREATE_OS2_GUEST_WITHOUT_FDD_IMAGE Macro

Syntax

```
#define PET_QUESTION_CREATE_OS2_GUEST_WITHOUT_FDD_IMAGE PRL_RESULT_DECLARE_SUCCESS(13020)
```

File

PrlErrorsValues.h

PET_QUESTION_CREATE_VM_FROM_LION_RECOVERY_PART Macro

Syntax

```
#define PET_QUESTION_CREATE_VM_FROM_LION_RECOVERY_PART PRL_RESULT_DECLARE_SUCCESS(13052)
```

File

PrlErrorsValues.h

PET_QUESTION_DELAYED_COMPACT_VM_DISKS Macro

Syntax

```
#define PET_QUESTION_DELAYED_COMPACT_VM_DISKS PRL_RESULT_DECLARE_SUCCESS(13032)
```

File

PrlErrorsValues.h

PET_QUESTION_DELETE_FILES_OUT_OF_VM_DIR Macro

Syntax

```
#define PET_QUESTION_DELETE_FILES_OUT_OF_VM_DIR PRL_RESULT_DECLARE_SUCCESS(13006)
```

File

PrlErrorsValues.h

PET_QUESTION_DELETE_PARALLELS_FILES_IN_VM_DIR Macro

Syntax

```
#define PET_QUESTION_DELETE_PARALLELS_FILES_IN_VM_DIR PRL_RESULT_DECLARE_SUCCESS(13007)
```

File

PrlErrorsValues.h

PET_QUESTION_DELETE_VM_WITH_CORRUPTED_CONFIG Macro

Syntax

```
#define PET_QUESTION_DELETE_VM_WITH_CORRUPTED_CONFIG PRL_RESULT_DECLARE_SUCCESS(13009)
```

File

PrlErrorsValues.h

PET_QUESTION_DO_YOU_WANT_TO_OVERWRITE_FILE Macro

Syntax

```
#define PET_QUESTION_DO_YOU_WANT_TO_OVERWRITE_FILE PRL_RESULT_DECLARE_SUCCESS(13002)
```

File

PrlErrorsValues.h

PET_QUESTION_FORCE_COMPACT_VM_DISKS Macro

Syntax

```
#define PET_QUESTION_FORCE_COMPACT_VM_DISKS PRL_RESULT_DECLARE_SUCCESS(13034)
```

File

PrlErrorsValues.h

PET_QUESTION_FREE_SIZE_FOR_COMPRESSED_DISK Macro

Syntax

```
#define PET_QUESTION_FREE_SIZE_FOR_COMPRESSED_DISK PRL_RESULT_DECLARE_SUCCESS(13005)
```

File

PrlErrorsValues.h

PET_QUESTION_MAC_PHYSICAL_MEMORY_MAP_BUG Macro

Syntax

```
#define PET_QUESTION_MAC_PHYSICAL_MEMORY_MAP_BUG PRL_RESULT_DECLARE_SUCCESS(13019)
```

File

PrlErrorsValues.h

PET_QUESTION_OLD_CONFIG_CONVERTION Macro

Syntax

```
#define PET_QUESTION_OLD_CONFIG_CONVERTION PRL_RESULT_DECLARE_SUCCESS(13004)
```

File

PrlErrorsValues.h

PET_QUESTION_ON_QUERY_END_SESSION Macro

Syntax

```
#define PET_QUESTION_ON_QUERY_END_SESSION PRL_RESULT_DECLARE_SUCCESS(13038)
```

File

PrlErrorsValues.h

PET_QUESTION_ON_QUERY_END_SESSION_RESTRICTED Macro

Syntax

```
#define PET_QUESTION_ON_QUERY_END_SESSION_RESTRICTED PRL_RESULT_DECLARE_SUCCESS(13039)
```

File

PrlErrorsValues.h

PET_QUESTION_REACH_OVERCOMMIT_STATE Macro

Syntax

```
#define PET_QUESTION_REACH_OVERCOMMIT_STATE PRL_RESULT_DECLARE_SUCCESS(13014)
```

File

PrlErrorsValues.h

PET_QUESTION_REACH_OVERCOMMIT_STATE_ON_SETMEM Macro

Syntax

```
#define PET_QUESTION_REACH_OVERCOMMIT_STATE_ON_SETMEM PRL_RESULT_DECLARE_SUCCESS(13017)
```

File

PrlErrorsValues.h

PET_QUESTION_REBOOT_HOST_ON_PCI_DRIVER_INSTALL_OR_REVERT Macro

Syntax

```
#define PET_QUESTION_REBOOT_HOST_ON_PCI_DRIVER_INSTALL_OR_REVERT
PRL_RESULT_DECLARE_SUCCESS(13027)
```

File

PrlErrorsValues.h

PET_QUESTION_REGISTER_USED_VM Macro

Syntax

```
#define PET_QUESTION_REGISTER_USED_VM PRL_RESULT_DECLARE_SUCCESS(13010)
```

File

PrlErrorsValues.h

PET_QUESTION_REGISTER_VM_TEMPLATE Macro

Syntax

```
#define PET_QUESTION_REGISTER_VM_TEMPLATE PRL_RESULT_DECLARE_SUCCESS(13008)
```

File

PrlErrorsValues.h

PET_QUESTION_RESTART_VM_GUEST_TO_COMPACT Macro

Syntax

```
#define PET_QUESTION_RESTART_VM_GUEST_TO_COMPACT PRL_RESULT_DECLARE_SUCCESS(13029)
```

File

PrlErrorsValues.h

PET_QUESTION_RESTORE_VM_CONFIG_FROM_BACKUP Macro

Syntax

```
#define PET_QUESTION_RESTORE_VM_CONFIG_FROM_BACKUP PRL_RESULT_DECLARE_SUCCESS(13023)
```

File

PrlErrorsValues.h

PET_QUESTION_SAMPLE_1 Macro

- Questions

+++ ALIGNED VALUE TO 13000

Syntax

```
#define PET_QUESTION_SAMPLE_1 PRL_RESULT_DECLARE_SUCCESS(13000)
```

File

PrlErrorsValues.h

PET_QUESTION_SAMPLE_2 Macro

Syntax

```
#define PET_QUESTION_SAMPLE_2 PRL_RESULT_DECLARE_SUCCESS(13001)
```

File

PrlErrorsValues.h

PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE Macro

Syntax

```
#define PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE PRL_RESULT_DECLARE_SUCCESS(13054)
```

File

PrlErrorsValues.h

PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE_CPU Macro

Syntax

```
#define PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE_CPU PRL_RESULT_DECLARE_SUCCESS(13051)
```

File

PrlErrorsValues.h

PET_QUESTION_STOP_VM_TO_COMPACT Macro

Syntax

```
#define PET_QUESTION_STOP_VM_TO_COMPACT PET_QUESTION_RESTART_VM_GUEST_TO_COMPACT
```

File

PrlErrorsValues.h

PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE Macro

Syntax

```
#define PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE PRL_RESULT_DECLARE_SUCCESS(13053)
```

File

PrlErrorsValues.h

PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE_CPU Macro

Syntax

```
#define PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE_CPU PRL_RESULT_DECLARE_SUCCESS(13050)
```

File

PrlErrorsValues.h

PET_QUESTION_SWITCH_OFF_AUTO_COMPRESS Macro

Syntax

```
#define PET_QUESTION_SWITCH_OFF_AUTO_COMPRESS PRL_RESULT_DECLARE_SUCCESS(13033)
```

File

PrlErrorsValues.h

PET_QUESTION_TXT_ERROR Macro

Syntax

```
#define PET_QUESTION_TXT_ERROR PRL_RESULT_DECLARE_SUCCESS(13003)
```

File

PrlErrorsValues.h

PET_QUESTION_UNDO_DISKS_MODE Macro

Syntax

```
#define PET_QUESTION_UNDO_DISKS_MODE PRL_RESULT_DECLARE_SUCCESS(13018)
```

File

PrlErrorsValues.h

PET_QUESTION_VM_COPY_OR_MOVE Macro

Syntax

```
#define PET_QUESTION_VM_COPY_OR_MOVE PRL_RESULT_DECLARE_SUCCESS(13013)
```

File

PrlErrorsValues.h

PET_QUESTION_VM_REBOOT_REQUIRED_BY_PRL_TOOLS Macro

Syntax

```
#define PET_QUESTION_VM_REBOOT_REQUIRED_BY_PRL_TOOLS PRL_RESULT_DECLARE_SUCCESS(13041)
```

File

PrlErrorsValues.h

PET_QUESTION_VM_ROOT_DIRECTORY_NOT_EXISTS Macro

Syntax

```
#define PET_QUESTION_VM_ROOT_DIRECTORY_NOT_EXISTS PRL_RESULT_DECLARE_SUCCESS(13028)
```

File

PrlErrorsValues.h

PRL_DISK Macros

Macros

Name	Description
PRL_DISK_ALIGN_REQUESTS (p. 1129)	Incoming requests to read/write async must be gathered and aligned
PRL_DISK_ALLOW_DATA_MODIFY (p. 1129)	Allow inplace data modification to reduce memory copying and allocation
PRL_DISK_CACHE_DEFAULT (p. 1130)	Disk cache default.
PRL_DISK_CACHE_DISABLED (p. 1130)	Uncached access to disk image.
PRL_DISK_CACHE_ENABLED (p. 1130)	Cached access to disk image.
PRL_DISK_CACHE_UNCACHED_READ (p. 1130)	Cached write/uncached read.

PRL_DISK_CACHE_UNCACHED_SEQ_READ (p. 1131)	Like PRL_DISK_CACHE_UNCACHED_READ (p. 1130), but cache is enabled for • small and random/reordered requests.
PRL_DISK_DEFAULT_FLAG (p. 1131)	Default (writelock always on at write on).
PRL_DISK_FAKE_OPEN (p. 1131)	Fake open, just for parameters get (processed by real disks that need administrative privileges to work with).
PRL_DISK_FOPS_DEFAULT (p. 1132)	Default access mode that compiled in Virtuozzo itself.
PRL_DISK_FOPS_MASK (p. 1132)	Type of the file access to use (15 file accessors is enough).
PRL_DISK_FOPS_MERGING (p. 1132)	DEPRECATED! Access mode with requests merging.
PRL_DISK_FOPS_NONCACHED (p. 1132)	DEPRECATED! Access mode with file caching turned off.
PRL_DISK_FOPS_NORMAL (p. 1133)	DEPRECATED! Normal access mode (default os)
PRL_DISK_FOPS_PCS (p. 1133)	DEPRECATED! Access mode to files on PCS storage
PRL_DISK_IGNORE_PROTECTION (p. 1133)	Ignore storages read only flag (useful at disk creation)
PRL_DISK_INVALID (p. 1133)	Last (invalid).
PRL_DISK_NO_ERROR_CHECKING (p. 1134)	Force open of disk for reading without error checking
PRL_DISK_NO_SPACE_CHECKS (p. 1134)	Do not check host free space
PRL_DISK_NOT_LIMIT_HANDLES (p. 1134)	Do not use handle pool for disk image handles
PRL_DISK_NOTHING (p. 1134)	Nothing.
PRL_DISK_PROGRESS_COMPLETED (p. 1135)	
PRL_DISK_PROGRESS_MAX (p. 1135)	
PRL_DISK_PROGRESS_MIN (p. 1135)	States progress definition.
PRL_DISK_READ (p. 1135)	Read
PRL_DISK_READ_LOCK (p. 1135)	ReadLock
PRL_DISK_USE_RAW_DISK (p. 1136)	Use raw disk for BootCamp partition (/dev/rdisk instead of /dev/disk)
PRL_DISK_WRITE (p. 1136)	Write
PRL_DISK_XML_CHANGE (p. 1136)	This parameter is used when user need to change XML data but not access to the drive itself.

PRL_DISK_ALIGN_REQUESTS Macro

Incoming requests to read/write async must be gathered and aligned

Syntax

```
#define PRL_DISK_ALIGN_REQUESTS 0x00020000
```

File

PrlDisk.h

PRL_DISK_ALLOW_DATA_MODIFY Macro

Allow inplace data modification to reduce memory copying and allocation

Syntax

```
#define PRL_DISK_ALLOW_DATA_MODIFY 0x00010000
```

File

PrlDisk.h

PRL_DISK_CACHE_DEFAULT Macro

Disk cache default.

Syntax

```
#define PRL_DISK_CACHE_DEFAULT 0
```

File

PrlDisk.h

PRL_DISK_CACHE_DISABLED Macro

Uncached access to disk image.

Syntax

```
#define PRL_DISK_CACHE_DISABLED 2
```

File

PrlDisk.h

Remarks

Slow read/write always, but host OS cache is not used, so it's good for host applications.

PRL_DISK_CACHE_ENABLED Macro

Cached access to disk image.

Syntax

```
#define PRL_DISK_CACHE_ENABLED 1
```

File

PrlDisk.h

Remarks

Fast write() and read() in case of all data can be fit in OS cache. But in the other hand host cache will be partially duplicated in guest OS cache, so in this mode memory usage may be ineffective.

PRL_DISK_CACHE_UNCACHED_READ Macro

Cached write/uncached read.

Syntax

```
#define PRL_DISK_CACHE_UNCACHED_READ 3
```

File

PrlDisk.h

Remarks

Like DISK_CACHE_ENABLED, but read data does not duplicated in host OS RAM, so you have more RAM for host applications. On Mac you may also see performance gain when you read data that were just written before and held in OS cache - data will be read from cache even with F_NOCACHE read(). This may help guest OS with stuff like pagefile.sys for example.

PRL_DISK_CACHE_UNCACHED_SEQ_READ Macro

Like PRL_DISK_CACHE_UNCACHED_READ (p. 1130), but cache is enabled for

- small and random/reordered requests.

Syntax

```
#define PRL_DISK_CACHE_UNCACHED_SEQ_READ 4
```

File

PrlDisk.h

Remarks

This let fix some

- performance problems when guest read data sequentially, but
- by small buffers.

PRL_DISK_DEFAULT_FLAG Macro

Default (writelock always on at write on).

Syntax

```
#define PRL_DISK_DEFAULT_FLAG (PRL_DISK_READ | PRL_DISK_WRITE | PRL_DISK_FOPS_DEFAULT)
```

File

PrlDisk.h

PRL_DISK_FAKE_OPEN Macro

Fake open, just for parameters get (processed by real disks that need administrative privileges to work with).

Syntax

```
#define PRL_DISK_FAKE_OPEN 0x00000008
```

File

PrlDisk.h

PRL_DISK_FOPS_DEFAULT Macro

Default access mode that compiled in Virtuozzo itself.

Syntax

```
#define PRL_DISK_FOPS_DEFAULT 0x00000000
```

File

PrlDisk.h

PRL_DISK_FOPS_MASK Macro

Type of the file access to use (15 file accessors is enough).

Syntax

```
#define PRL_DISK_FOPS_MASK 0x000003C0
```

File

PrlDisk.h

PRL_DISK_FOPS_MERGING Macro

DEPRECATED! Access mode with requests merging.

Syntax

```
#define PRL_DISK_FOPS_MERGING 0x000000C0
```

File

PrlDisk.h

PRL_DISK_FOPS_NONCACHED Macro

DEPRECATED! Access mode with file caching turned off.

Syntax

```
#define PRL_DISK_FOPS_NONCACHED 0x00000080
```

File

PrlDisk.h

PRL_DISK_FOPS_NORMAL Macro

DEPRECATED! Normal access mode (default os)

Syntax

```
#define PRL_DISK_FOPS_NORMAL 0x00000040
```

File

PrlDisk.h

PRL_DISK_FOPS_PCS Macro

DEPRECATED! Access mode to files on PCS storage

Syntax

```
#define PRL_DISK_FOPS_PCS 0x000001C0
```

File

PrlDisk.h

PRL_DISK_IGNORE_PROTECTION Macro

Ignore storages read only flag (useful at disk creation)

Syntax

```
#define PRL_DISK_IGNORE_PROTECTION 0x00040000
```

File

PrlDisk.h

PRL_DISK_INVALID Macro

Last (invalid).

Syntax

```
#define PRL_DISK_INVALID
```

File

PrlDisk.h

PRL_DISK_NO_ERROR_CHECKING Macro

Force open of disk for reading without error checking

Syntax

```
#define PRL_DISK_NO_ERROR_CHECKING 0x00000400
```

File

PrlDisk.h

PRL_DISK_NO_SPACE_CHECKS Macro

Do not check host free space

Syntax

```
#define PRL_DISK_NO_SPACE_CHECKS 0x00008000
```

File

PrlDisk.h

PRL_DISK_NOT_LIMIT_HANDLES Macro

Do not use handle pool for disk image handles

Syntax

```
#define PRL_DISK_NOT_LIMIT_HANDLES 0x00004000
```

File

PrlDisk.h

PRL_DISK_NOTHING Macro

Nothing.

Syntax

```
#define PRL_DISK_NOTHING 0x00000000
```

File

PrlDisk.h

PRL_DISK_PROGRESS_COMPLETED Macro

Syntax

```
#define PRL_DISK_PROGRESS_COMPLETED 1005
```

File

PrlDisk.h

PRL_DISK_PROGRESS_MAX Macro

Syntax

```
#define PRL_DISK_PROGRESS_MAX 1000
```

File

PrlDisk.h

PRL_DISK_PROGRESS_MIN Macro

States progress definition.

Syntax

```
#define PRL_DISK_PROGRESS_MIN 0
```

File

PrlDisk.h

PRL_DISK_READ Macro

Read

Syntax

```
#define PRL_DISK_READ 0x00000001
```

File

PrlDisk.h

PRL_DISK_READ_LOCK Macro

ReadLock

Syntax

```
#define PRL_DISK_READ_LOCK 0x00000004
```

File

PrlDisk.h

PRL_DISK_USE_RAW_DISK Macro

Use raw disk for BootCamp partition (/dev/rdisk instead of /dev/disk)

Syntax

```
#define PRL_DISK_USE_RAW_DISK 0x00002000
```

File

PrlDisk.h

PRL_DISK_WRITE Macro

Write

Syntax

```
#define PRL_DISK_WRITE 0x00000002
```

File

PrlDisk.h

PRL_DISK_XML_CHANGE Macro

This parameter is used when user need to change XML data but not access to the drive itself.

Syntax

```
#define PRL_DISK_XML_CHANGE 0x00000020
```

File

PrlDisk.h

Remarks

It can prevent situations when VirtualDisk library should unmount drive from host system to store small amount of data to the XML descriptor.

PRL_ERR Macros

Macros

Name	Description
PRL_ERR_ACCES_DENIED_FILE_TO_PARENT_PARENT_DIR (p. 1175)	
PRL_ERR_ACCESS_DENIED (p. 1175)	
PRL_ERR_ACCESS_DENIED_TO_CHANGE_PERMISSIONS (p. 1175)	
PRL_ERR_ACCESS_DENIED_TO_DISK_IMAGE (p. 1176)	
PRL_ERR_ACCESS_DENINED_TO_RUN_WIZARD (p. 1176)	
PRL_ERR_ACCESS_TO_CLONE_VM_DEVICE_DENIED (p. 1176)	
PRL_ERR_ACCESS_TO_VM_DENIED (p. 1176)	
PRL_ERR_ACCESS_TO_VM_HDD_DENIED (p. 1176)	
PRL_ERR_ACCESS_TOKEN_INVALID (p. 1177)	
PRL_ERR_ACTION_NOT_SUPPORTED_FOR_CT (p. 1177)	
PRL_ERR_ACTIVATE_NO_INSTALLED_LICENSE (p. 1177)	
PRL_ERR_ACTIVATE_TRIAL_LICENSE (p. 1177)	
PRL_ERR_ACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE (p. 1178)	
PRL_ERR_ACTIVATION_COMMON_SERVER_ERROR (p. 1178)	
PRL_ERR_ACTIVATION_HTTP_REQUEST_FAILED (p. 1178)	
PRL_ERR_ACTIVATION_OFFLINE_PERIOD_EXPIRED (p. 1178)	
PRL_ERR_ACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID (p. 1179)	
PRL_ERR_ACTIVATION_SERVER_ERROR (p. 1179)	
PRL_ERR_ACTIVATION_SERVER_HWIDS_AMOUNT_REACHED (p. 1179)	
PRL_ERR_ACTIVATION_SERVER_KEY_IS_BLACKLISTED (p. 1179)	
PRL_ERR_ACTIVATION_SERVER_KEY_IS_INVALID (p. 1180)	
PRL_ERR_ACTIVATION_UNABLE_TO_SEND_REQUEST (p. 1180)	
PRL_ERR_ACTIVATION_UPDATE_FAILED (p. 1180)	
PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_FORMAT (p. 1180)	• license activation errors * +++ ALIGNED VALUE TO 11700
PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_MESSAGE (p. 1181)	
PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_SIGNATURE (p. 1181)	

PRL_ERR_ACTIVATION_WRONG_SERVER_RESPONSE (p. 1181)	
PRL_ERR_ADD_HW_FLOPPY_IMAGE_NOT_SPECIFYED (p. 1181)	
PRL_ERR_ADD_VM_OPERATION_WAS_CANCELED (p. 1182)	
PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_OPERATION (p. 1182)	
PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_VM_OPERATION (p. 1182)	
PRL_ERR_ALIGN_ON_4 (p. 1182)	
PRL_ERR_ALREADY_CONNECTED_TO_DISPATCHER (p. 1183)	
PRL_ERR_ANOTHER_USER_SESSIONS_PRESENT (p. 1183)	Dispatcher service mode errors codes
PRL_ERR_API_INCOMPATIBLE (p. 1183)	
PRL_ERR_API_WASNT_INITIALIZED (p. 1183)	
PRL_ERR_APPLIANCE_CANNOT_CALCULATE_MD5 (p. 1183)	
PRL_ERR_APPLIANCE_CANNOT_CHANGE_OWNER (p. 1184)	
PRL_ERR_APPLIANCE_CANNOT_EXTRACT_VM (p. 1184)	
PRL_ERR_APPLIANCE_CANNOT_MOVE_VM_BUNDLE (p. 1184)	
PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_DIR (p. 1184)	
PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_EXISTS (p. 1185)	
PRL_ERR_APPLIANCE_DOWNLOAD_PATH_CANNOT_CREATE (p. 1185)	
PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_FOUND (p. 1185)	
PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_STARTED (p. 1185)	
PRL_ERR_APPLIANCE_EXIT_WITH_ERROR (p. 1186)	
PRL_ERR_APPLIANCE_INSTALL_ALREADY_IN_PROCESS (p. 1186)	
PRL_ERR_APPLIANCE_INSTALL_WAS_NOT_STARTED (p. 1186)	
PRL_ERR_APPLIANCE_INVALID_CONFIG (p. 1186)	Install appliance task errors +++ ALIGNED VALUE TO 41500
PRL_ERR_APPLIANCE_MISMATCH_MD5 (p. 1187)	
PRL_ERR_APPLIANCE_NAME_NOT_MATCH_VM_BUNDLE (p. 1187)	
PRL_ERR_APPLIANCE_USER_NOT_FOUND (p. 1187)	
PRL_ERR_ATTACH_BACKUP_ALREADY_ATTACHED (p. 1187)	
PRL_ERR_ATTACH_BACKUP_BUSE_NOT_MOUNTED (p. 1187)	
PRL_ERR_ATTACH_BACKUP_CUSTOM_BACKUP_SERVER_NOT_SUPPORTED (p. 1188)	
PRL_ERR_ATTACH_BACKUP_FORMAT_NOT_SUPPORTED (p. 1188)	
PRL_ERR_ATTACH_BACKUP_INTERNAL_ERROR (p. 1188)	Attach backup as a block device errors +++ ALIGNED VALUE TO 54000
PRL_ERR_ATTACH_BACKUP_INVALID_STORAGE_URL (p. 1188)	
PRL_ERR_ATTACH_BACKUP_PROTO_ERROR (p. 1189)	
PRL_ERR_ATTACH_BACKUP_URL_CHANGE_PROHIBITED (p. 1189)	
PRL_ERR_ATTACH_TO_TASK_BAD_SESSION (p. 1189)	
PRL_ERR_AUDIO_ENCODINGS_NOT_SUPPORTED (p. 1189)	
PRL_ERR_AUTHENTICATION_FAILED (p. 1190)	
PRL_ERR_BACKUP_ACCESS_TO_VM_DENIED (p. 1190)	

PRL_ERR_BACKUP_ACRONIS_ERR (p. 1190)	
PRL_ERR_BACKUP_BACKUP_CMD_FAILED (p. 1190)	
PRL_ERR_BACKUP_BACKUP_NOT_FOUND (p. 1190)	
PRL_ERR_BACKUP_BACKUP_UUID_NOT_FOUND (p. 1191)	
PRL_ERR_BACKUP_CANNOT_CREATE_DIRECTORY (p. 1191)	
PRL_ERR_BACKUP_CANNOT_REMOVE_DIRECTORY (p. 1191)	
PRL_ERR_BACKUP_CANNOT_SET_PERMISSIONS (p. 1191)	
PRL_ERR_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE (p. 1192)	
PRL_ERR_BACKUP_CREATE_SNAPSHOT_FAILED (p. 1192)	
PRL_ERR_BACKUP_CT_ID_ALREADY_EXIST (p. 1192)	
PRL_ERR_BACKUP_DIRECTORY_ALREADY_EXIST (p. 1192)	
PRL_ERR_BACKUP_INTERNAL_ERROR (p. 1193)	
PRL_ERR_BACKUP_INTERNAL_PROTO_ERROR (p. 1193)	VM Backup errors +++ ALIGNED VALUE TO 37000
PRL_ERR_BACKUP_LOCKED_FOR_READING (p. 1193)	
PRL_ERR_BACKUP_LOCKED_FOR_WRITING (p. 1193)	
PRL_ERR_BACKUP_REGISTER_VM_FAILED (p. 1193)	
PRL_ERR_BACKUP_REMOVE_PERMISSIONS_DENIED (p. 1194)	
PRL_ERR_BACKUP_REQUIRE_LOGIN_PASSWORD (p. 1194)	
PRL_ERR_BACKUP_RESTORE_CANNOT_CREATE_DIRECTORY (p. 1194)	
PRL_ERR_BACKUP_RESTORE_CMD_FAILED (p. 1194)	
PRL_ERR_BACKUP_RESTORE_DIRECTORY_ALREADY_EXIST (p. 1195)	
PRL_ERR_BACKUP_RESTORE_EXISTING_SET_PRIVATE (p. 1195)	
PRL_ERR_BACKUP_RESTORE_INTERNAL_ERROR (p. 1195)	
PRL_ERR_BACKUP_RESTORE_INTERNAL_PROTO_ERROR (p. 1195)	
PRL_ERR_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE (p. 1196)	
PRL_ERR_BACKUP_RESTORE_PROHIBIT_WHEN_ATTACHED (p. 1196)	
PRL_ERR_BACKUP_RESTORE_VM_RUNNING (p. 1196)	
PRL_ERR_BACKUP_SNAPSHOT_OF_PAUSED_VM (p. 1196)	
PRL_ERR_BACKUP_SWITCH_TO_SNAPSHOT_FAILED (p. 1197)	
PRL_ERR_BACKUP_TIMEOUT_EXCEEDED (p. 1197)	
PRL_ERR_BACKUP_TOOL_CANNOT_START (p. 1197)	
PRL_ERR_BACKUP_UNSUPPORTED_KEEP_CHAIN (p. 1197)	
PRL_ERR_BACKUP_UNSUPPORTED_STORAGE_VERSION (p. 1198)	
PRL_ERR_BAD_DISP_CONFIG_FILE_SPECIFIED (p. 1198)	
PRL_ERR_BAD_PARAMETERS (p. 1198)	
PRL_ERR_BAD_VM_CONFIG_FILE_SPECIFIED (p. 1198)	
PRL_ERR_BAD_VM_DIR_CONFIG_FILE_SPECIFIED (p. 1198)	
PRL_ERR_BRIDGE_NOT_FOUND_FOR_NETWORK_ADAPTER (p. 1199)	
PRL_ERR_BUFFER_OVERRUN (p. 1199)	
PRL_ERR_BUSE_ENTRY_ALREADY_EXIST (p. 1199)	
PRL_ERR_BUSE_ENTRY_ALREADY_INITIALIZED (p. 1199)	
PRL_ERR_BUSE_ENTRY_INVALID (p. 1200)	
PRL_ERR_BUSE_ENTRY_IO_ERROR (p. 1200)	
PRL_ERR_BUSE_INTERNAL_ERROR (p. 1200)	

PRL_ERR_BUSE_NOT_MOUNTED (p. 1200)	BUSE filesystem errors +++ ALIGNED VALUE TO 52000
PRL_ERR_CAN_NOT_GET_DISK_FREE_SPACE (p. 1200)	
PRL_ERR_CANNOT_EDIT_FIREWALL_AT_TRANS_VM_STATE (p. 1201)	
PRL_ERR_CANNOT_EDIT_HARDWARE_FOR_PAUSED_VM (p. 1201)	
PRL_ERR_CANNOT_PROCESSING_SAFE_MODE_FOR_INVALID_VM (p. 1201)	
PRL_ERR_CANNOT_PROCESSING_UNDO_DISKS_FOR_INVALID_VM (p. 1202)	
PRL_ERR_CANNOT_SAVE_REMOTE_DEVICE_STATE (p. 1202)	
PRL_ERR_CANT_ALOCA_MEM_FILE_COPY (p. 1202)	
PRL_ERR_CANT_CHANGE_FILE_PERMISSIONS (p. 1202)	
PRL_ERR_CANT_CHANGE_HOST_ID (p. 1203)	
PRL_ERR_CANT_CHANGE_OWNER_OF_DISK_IMAGE_FILE (p. 1203)	
PRL_ERR_CANT_CHANGE_OWNER_OF_FILE (p. 1203)	
PRL_ERR_CANT_CHANGE_OWNER_OF_VM_FILE (p. 1203)	
PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL (p. 1203)	
PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL_BY_NON_PRIVILEGED_USER (p. 1204)	
PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN (p. 1204)	
PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN_BY_NON_PRIVILEGED_USER (p. 1204)	
PRL_ERR_CANT_CONFIGURE_PARTITION_HDD (p. 1204)	
PRL_ERR_CANT_CONFIGURE_PHYSICAL_HDD (p. 1205)	
PRL_ERR_CANT_CONNECT_TO_DISPATCHER (p. 1205)	
PRL_ERR_CANT_CONNECT_TO_DISPATCHER_ITERATIVELY (p. 1205)	
PRL_ERR_CANT_CONVERT_CONFIG (p. 1205)	
PRL_ERR_CANT_CONVERT_HDD_IMAGE_FILE (p. 1206)	
PRL_ERR_CANT_CONVERT_OLD_VM_WITH_INSTALLED_TOOLS (p. 1206)	
PRL_ERR_CANT_CONVERT_OTHER_VENDOR_HDD (p. 1206)	
PRL_ERR_CANT_CONVERT_OTHER_VENDOR_VM (p. 1206)	
PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_DISKS_PRESENT (p. 1207)	
PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_SNAPSHOTS_PRESENT (p. 1207)	
PRL_ERR_CANT_CREATE_FLOPPY_IMAGE (p. 1207)	
PRL_ERR_CANT_CREATE_HDD_IMAGE (p. 1207)	
PRL_ERR_CANT_CREATE_HDD_IMAGE_NO_SPACE (p. 1208)	
PRL_ERR_CANT_CREATE_PARALLEL_PORT_IMAGE (p. 1208)	
PRL_ERR_CANT_CREATE_SERIAL_PORT_IMAGE (p. 1208)	
PRL_ERR_CANT_DELETE_FILE (p. 1208)	
PRL_ERR_CANT_EDIT_SUSPENDED_VM (p. 1208)	
PRL_ERR_CANT_GET_FILE_PERMISSIONS (p. 1209)	
PRL_ERR_CANT_INIT_REAL_CPUS_INFO (p. 1209)	
PRL_ERR_CANT_PARSE_DISP_EVENT (p. 1209)	
PRL_ERR_CANT_PARSE_VM_CONFIG (p. 1209)	
PRL_ERR_CANT_PREPARE_RECONFIG_DATA (p. 1209)	
PRL_ERR_CANT_RECONFIG_GUEST_OS (p. 1210)	
PRL_ERR_CANT_RECREATE_HDD (p. 1210)	

PRL_ERR_CANT_REMOVE_DIR_ENTRY (p. 1210)	
PRL_ERR_CANT_REMOVE_ENTRY (p. 1210)	
PRL_ERR_CANT_REMOVE_INVALID_VM_AS_NON_ADMIN (p. 1210)	
PRL_ERR_CANT_RENAME_DIR_ENTRY (p. 1211)	
PRL_ERR_CANT_RENAME_ENTRY (p. 1211)	
PRL_ERR_CANT_REPLACE_FLOPPY_IMAGE (p. 1211)	
PRL_ERR_CANT_RESIZE_HDD (p. 1211)	
PRL_ERR_CANT_RESOLVE_HOSTNAME (p. 1212)	
PRL_ERR_CANT_REVERT_VM_SINCE_VTD_DEVICE_ALREADY_USED (p. 1212)	
PRL_ERR_CANT_START_VM_SINCE_NO_VTD_DRIVER (p. 1212)	
PRL_ERR_CANT_START_VM_SINCE_VTD_DEVICE_ALREADY_USED (p. 1212)	
PRL_ERR_CANT_SUSPEND_VM_WITH_BOOTCAMP (p. 1213)	
PRL_ERR_CANT_TO_CHANGE_PERMISSIONS_ON_REMOTE_LOCATION (p. 1213)	
PRL_ERR_CANT_TO_START_VM_TEMPLATE (p. 1213)	
PRL_ERR_CANT_UNPACK_ARCHIVE (p. 1213)	
PRL_ERR_CANT_UPDATE_DEVICE_INFO (p. 1214)	
PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT (p. 1214)	
PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT32 (p. 1214)	
PRL_ERR_CDD_IMAGE_NOT_SPECIFIED (p. 1214)	
PRL_ERR_CHANGESID_FAILED (p. 1214)	Vm changeSID errors +++ ALIGNED VALUE TO 41200
PRL_ERR_CHANGESID_GUEST_TOOLS_NOT_AVAILABLE (p. 1215)	
PRL_ERR_CHANGESID_NOT_AVAILABLE (p. 1215)	
PRL_ERR_CHANGESID_NOT_SUPPORTED (p. 1215)	
PRL_ERR_CHANGESID_VM_START_FAILED (p. 1215)	
PRL_ERR_CI_CANNOT_COPY_IMAGE_NON_STOPPED_VM (p. 1216)	
PRL_ERR_CI_DEVICE_IS_NOT_VIRTUAL (p. 1216)	Copy device image errors +++ ALIGNED VALUE TO 46700
PRL_ERR_CI_PERMISSIONS_DENIED (p. 1216)	
PRL_ERR_CLONE_OPERATION_CANCELED (p. 1216)	
PRL_ERR_CLUSTER_RESOURCE_ERROR (p. 1216)	Cluster resource manipulation errors +++ ALIGNED VALUE TO 46900
PRL_ERR_COM_DEVICE_ALREADY_EXIST (p. 1217)	
PRL_ERR_COM_OUTPUT_FILE_IS_NOT_SPECIFIED (p. 1217)	
PRL_ERR_COM_OUTPUT_FILE_NOT_EXIST (p. 1217)	
PRL_ERR_COMMAND_SUPPORTED_ONLY_AT_SERVER_MODE (p. 1217)	
PRL_ERR_COMMON_SERVER_PREFS_BLOCKED_TO_CHANGE (p. 1218)	
PRL_ERR_COMMON_SERVER_PREFS_WERE_CHANGED (p. 1218)	
PRL_ERR_COMPACT_ALREADY_SWITCHED_ON (p. 1218)	
PRL_ERR_COMPACT_WRONG_VM_STATE (p. 1218)	
PRL_ERR_CONCURRENT_COMMAND_EXECUTING (p. 1219)	
PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_OBJECT_UUID (p. 1219)	
PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_USER_UUID (p. 1219)	
PRL_ERR_CONFIG_EDIT_COLLISION (p. 1219)	
PRL_ERR_CONFIG_FILE_NOT_SET (p. 1220)	

PRL_ERR_CONFIGURE_GENERIC_PCI_TASK_ALREADY_RUN (p. 1220)	
PRL_ERR_CONFIRMATION_MODE_ALREADY_DISABLED (p. 1220)	
PRL_ERR_CONFIRMATION_MODE_ALREADY_ENABLED (p. 1220)	
PRL_ERR_CONFIRMATION_MODE_UNABLE_CHANGE_BY_NOT_ADMIN (p. 1221)	confirmation errors +++ ALIGNED VALUE TO 41100
PRL_ERR_CONN_CLIENT_CERTIFICATE_EXPIRED (p. 1221)	
PRL_ERR_CONN_CLIENT_CERTIFICATE_INVALID (p. 1221)	
PRL_ERR_CONN_CLIENT_CERTIFICATE_REVOKED (p. 1221)	
PRL_ERR_CONN_SERVER_CERTIFICATE_EXPIRED (p. 1222)	
PRL_ERR_CONN_SERVER_CERTIFICATE_INVALID (p. 1222)	
PRL_ERR_CONN_SERVER_CERTIFICATE_REVOKED (p. 1222)	
PRL_ERR_CONN_UNABLE_TO_ESTABLISH_TRUSTED_CHANNEL (p. 1222)	
PRL_ERR_CONNECT_TO_MOUNTER (p. 1223)	
PRL_ERR_CONV_HD_CONFLICT (p. 1223)	
PRL_ERR_CONV_HD_DISK_TOOL_NOT_STARTED (p. 1223)	
PRL_ERR_CONV_HD_EXIT_WITH_ERROR (p. 1223)	
PRL_ERR_CONV_HD_NO_ONE_DISK_FOR_CONVERSION (p. 1223)	
PRL_ERR_CONV_HD_WRONG_VM_STATE (p. 1224)	Convert disks task errors +++ ALIGNED VALUE TO 41600
PRL_ERR_CONVERT_3RD_PARTY_VM_FAILED (p. 1224)	Vm convert errors +++ ALIGNED VALUE TO 41300
PRL_ERR_CONVERT_3RD_PARTY_VM_NO_SPACE (p. 1224)	
PRL_ERR_CONVERT_EFI_CONFIG_INVALID (p. 1224)	
PRL_ERR_CONVERT_EFI_GUEST_OS_UNSUPPORTED (p. 1225)	
PRL_ERR_CONVERT_GUEST_OS_IS_HIBERNATED (p. 1225)	
PRL_ERR_CONVERT_NO_GUEST_OS_FOUND (p. 1225)	
PRL_ERR_COPY_CT_TMPL_INTERNAL_ERROR (p. 1225)	
PRL_ERR_COPY_VM_INFO_FILE (p. 1225)	
PRL_ERR_CORE_STATE_CANCELLED (p. 1226)	
PRL_ERR_CORE_STATE_CHANGE_VM_CONFIG (p. 1226)	
PRL_ERR_CORE_STATE_CORRUPT_MEM_FILE (p. 1226)	
PRL_ERR_CORE_STATE_CORRUPT_SAV_FILE (p. 1226)	
PRL_ERR_CORE_STATE_ERROR_COMMON (p. 1226)	• Suspend/Resume codes +++ ALIGNED VALUE TO 20000
PRL_ERR_CORE_STATE_INV_SAV_VERSION (p. 1227)	
PRL_ERR_CORE_STATE_NO_FILE (p. 1227)	
PRL_ERR_CORE_STATE_VM_WOULD_RESTART (p. 1227)	
PRL_ERR_CORE_STATE_VM_WOULD_STOP (p. 1227)	
PRL_ERR_COULD_NOT_START_FIREWALL_TOOL (p. 1228)	
PRL_ERR_COULDNT_CREATE_AUTHORIZATION_FILE (p. 1228)	
PRL_ERR_COULDNT_SET_PERMISSIONS_TO_AUTHORIZATION_FILE (p. 1228)	
PRL_ERR_COULDNT_TO_CREATE_HDD_LINKED_CLONE (p. 1228)	
PRL_ERR_CPU_RESTART (p. 1229)	
PRL_ERR_CPU_SHUTDOWN (p. 1229)	

PRL_ERR_CPUFEATURES_DEFAULT_POOL_ARG (p. 1229)	
PRL_ERR_CPUFEATURES_INCOMPATIBLE_NODE (p. 1229)	
PRL_ERR_CPUFEATURES_INCORRECT_MASK (p. 1229)	
PRL_ERR_CPUFEATURES_NO_BINARY (p. 1230)	CPU features managment errors +++ ALIGNED VALUE TO 54100
PRL_ERR_CPUFEATURES_NOT_IN_POOLS (p. 1230)	
PRL_ERR_CPUFEATURES_POOLS_MANAGMENT (p. 1230)	
PRL_ERR_CPUFEATURES_RUNNING_VM_OR_CT (p. 1230)	
PRL_ERR_CREATE_BOOTABLE_ISO (p. 1230)	Unattended ISO errors +++ ALIGNED VALUE TO 40000
PRL_ERR_CREATE_HARD_DISK_WITH_ZERO_SIZE (p. 1231)	
PRL_ERR_CREATE_PRIVELEGED_PROCESS (p. 1231)	
PRL_ERR_CREATE_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT (p. 1231)	
PRL_ERR_CREATE_SNAPSHOT_VM_VTD_PAUSED (p. 1231)	
PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL (p. 1232)	
PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL (p. 1232)	
PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS (p. 1232)	
PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL (p. 1233)	
PRL_ERR_CT_IS_RUNNING (p. 1233)	
PRL_ERR_CT_MIGRATE_EXTERNAL_PROCESS (p. 1233)	
PRL_ERR_CT_MIGRATE_ID_ALREADY_EXIST (p. 1233)	
PRL_ERR_CT_MIGRATE_INTERNAL_ERROR (p. 1234)	
PRL_ERR_CT_MIGRATE_TARGET_ALREADY_EXISTS (p. 1234)	
PRL_ERR_CT_NOT_FOUND (p. 1234)	
PRL_ERR_CVSRC_IO_ERROR (p. 1234)	Captured Video Source errors +++ ALIGNED VALUE TO 46800
PRL_ERR_CVSRC_NO_CHANNEL (p. 1234)	
PRL_ERR_CVSRC_NO_OPEN_REQUEST (p. 1235)	
PRL_ERR_DEACTIVATE_NO_INSTALLED_LICENSE (p. 1235)	
PRL_ERR_DEACTIVATE_TRIAL_LICENSE (p. 1235)	
PRL_ERR_DEACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE (p. 1235)	
PRL_ERR_DEACTIVATION_COMMON_SERVER_ERROR (p. 1236)	
PRL_ERR_DEACTIVATION_HTTP_REQUEST_FAILED (p. 1236)	
PRL_ERR_DEACTIVATION_OLD_HWID_NOT_FOUND (p. 1236)	
PRL_ERR_DEACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID (p. 1236)	
PRL_ERR_DEACTIVATION_SERVER_ERROR (p. 1237)	
PRL_ERR_DEACTIVATION_SERVER_KEY_IS_BLACKLISTED (p. 1237)	
PRL_ERR_DEACTIVATION_SERVER_KEY_IS_INVALID (p. 1237)	
PRL_ERR_DEACTIVATION_UNABLE_TO_SEND_REQUEST (p. 1237)	
PRL_ERR_DEACTIVATION_WRONG_SERVER_RESPONSE (p. 1238)	
PRL_ERR_DELETE_UNFINISHED_STATE_FAILED (p. 1238)	
PRL_ERR_DELETING_VM_FROM_CATALOGUE (p. 1238)	

PRL_ERR_DEV_ALREADY_CONNECTED (p. 1238)	• Common Devices Errors +++ ALIGNED VALUE TO 9000
PRL_ERR_DEV_ALREADY_DISCONNECTED (p. 1239)	
PRL_ERR_DEV_DISABLE_AFTER_LOST (p. 1239)	
PRL_ERR_DEV_FLOPPY_CONNECT_FAILED (p. 1239)	ALIGNED VALUE TO 1000
PRL_ERR_DEV_MAX_CD_EXCEEDED_WITH_DISABLED_CLIENT_SCSII (p. 1239)	
PRL_ERR_DEV_MAX_CD_HDD_EXCEEDED (p. 1240)	
PRL_ERR_DEV_MAX_NUMBER_EXCEEDED (p. 1240)	• Other +++ ALIGNED VALUE TO 10000
PRL_ERR_DEV_PARALLEL_PORT_FILE_CONNECT_FAILED (p. 1240)	
PRL_ERR_DEV_PARALLEL_PORT_PHYSICAL_CONNECT_FAILED (p. 1240)	• Parallel Port +++ ALIGNED VALUE TO 7000
PRL_ERR_DEV_PARALLEL_PORT_PRINTER_CONNECT_FAILED (p. 1241)	
PRL_ERR_DEV_PARALLEL_PORT_REMOTE_CONNECT_FAILED (p. 1241)	
PRL_ERR_DEV_PRINTER_OVERFLOW (p. 1241)	
PRL_ERR_DEV_SERIAL_PORT_FILE_CONNECT_FAILED (p. 1241)	
PRL_ERR_DEV_SERIAL_PORT_PHYSICAL_CONNECT_FAILED (p. 1242)	• Serial Port +++ ALIGNED VALUE TO 6000
PRL_ERR_DEV_SERIAL_PORT_PIPE_CONNECT_FAILED (p. 1242)	
PRL_ERR_DEV_SERIAL_PORT_REMOTE_CONNECT_FAILED (p. 1242)	
PRL_ERR_DEV_USB_BUSY (p. 1242)	
PRL_ERR_DEV_USB_CHANGEPID (p. 1243)	
PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED (p. 1243)	
PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED2 (p. 1243)	
PRL_ERR_DEV_USB_INSTALL_DRIVER_FAILED (p. 1243)	
PRL_ERR_DEV_USB_NO_FREE_PORTS (p. 1244)	
PRL_ERR_DEV_USB_NOT_CONFIGURED (p. 1244)	
PRL_ERR_DEV_USB_OPEN_MANAGER_FAILED (p. 1244)	• USB +++ ALIGNED VALUE TO 8000
PRL_ERR_DEV_USB_REUSE (p. 1244)	
PRL_ERR_DIRECTORY_DOES_NOT_EXIST (p. 1244)	
PRL_ERR_DISK_BLOCK_CREATED (p. 1245)	
PRL_ERR_DISK_BLOCK_PARTIALLY_PROCESSED (p. 1245)	
PRL_ERR_DISK_BLOCK_SEARCH_FAILED (p. 1245)	
PRL_ERR_DISK_BLOCK_SKIPPED (p. 1245)	
PRL_ERR_DISK_BOOTCAMP_WRITE_MBR (p. 1246)	
PRL_ERR_DISK_CANT_INITIALIZE_IMAGE (p. 1246)	
PRL_ERR_DISK_COMPRESSED_FILE_EMPTY (p. 1246)	

PRL_ERR_DISK_CREATE_IMAGE_ERROR (p. 1246)	
PRL_ERR_DISK_DATA_NOT_FOUND (p. 1246)	• User parameters errors * +++ ALIGNED VALUE TO 23000
PRL_ERR_DISK_DIR_CREATE_ERROR (p. 1247)	
PRL_ERR_DISK_DISK_NOT_OPENED (p. 1247)	
PRL_ERR_DISK_ENLARGE_FAILED (p. 1247)	
PRL_ERR_DISK_FAT32_SIZE_EXCEEDED (p. 1247)	
PRL_ERR_DISK_FILE_CREATE_ERROR (p. 1248)	
PRL_ERR_DISK_FILE_EXISTS (p. 1248)	
PRL_ERR_DISK_FILE_OPEN_ERROR (p. 1248)	
PRL_ERR_DISK_FSYNC_FAILED (p. 1248)	
PRL_ERR_DISK_FULLFSYNC_FAILED (p. 1248)	
PRL_ERR_DISK_GENERIC_ERROR (p. 1249)	• Range reserved for different disk errors +++ ALIGNED VALUE TO 21000
PRL_ERR_DISK_GET_MOUNTPATH_FAILED (p. 1249)	• Bootcamp errors * +++ ALIGNED VALUE TO 24000
PRL_ERR_DISK_GET_PARAMS_FAILED (p. 1249)	
PRL_ERR_DISK_GET_SIZE_FAILED (p. 1249)	
PRL_ERR_DISK_GPT_MBR_NOT_EQUAL (p. 1250)	
PRL_ERR_DISK_GROUP_INTERSECT (p. 1250)	
PRL_ERR_DISK_IDENTIFY_FAILED (p. 1250)	
PRL_ERR_DISK_IMAGE_BUSY (p. 1250)	
PRL_ERR_DISK_IMAGES_CORRUPTED (p. 1250)	
PRL_ERR_DISK_IMPERSONATE_FAILED (p. 1251)	
PRL_ERR_DISK_INCORRECTLY_CLOSED (p. 1251)	
PRL_ERR_DISK_INSUFFICIENT_SPACE (p. 1251)	
PRL_ERR_DISK_INTERNAL_CLASS_ERROR (p. 1251)	
PRL_ERR_DISK_INVALID_BLOCKSIZE (p. 1251)	
PRL_ERR_DISK_INVALID_FORMAT (p. 1252)	
PRL_ERR_DISK_INVALID_PARAMETERS (p. 1252)	
PRL_ERR_DISK_MEMORY_ERROR (p. 1252)	
PRL_ERR_DISK_MOUNT_FAILED (p. 1252)	
PRL_ERR_DISK_MOUNTED_OVERLAP (p. 1252)	
PRL_ERR_DISK_NOT_IMPLEMENTED (p. 1253)	
PRL_ERR_DISK_NOT_PERMITTED (p. 1253)	
PRL_ERR_DISK_NOT_VALID_OFFSET (p. 1253)	
PRL_ERR_DISK_NULL_PART_SIZE (p. 1253)	
PRL_ERR_DISK_OFFSETS_FIXED (p. 1253)	
PRL_ERR_DISK_OPERATION_ABORTED (p. 1254)	
PRL_ERR_DISK_OPERATION_IN_PROGRESS (p. 1254)	

PRL_ERR_DISK_OPERATION_NOT_ALLOWED (p. 1254)	
PRL_ERR_DISK_PARTITION_INVALID_NAME (p. 1254)	
PRL_ERR_DISK_PARTITION_NOT_FOUND (p. 1254)	
PRL_ERR_DISK_PARTITIONS_TABLE_CYCLE (p. 1255)	
PRL_ERR_DISK_POINTERS_MIXED_UP (p. 1255)	
PRL_ERR_DISK_POSSIBLE_OVERRUN (p. 1255)	
PRL_ERR_DISK_READ_FAILED (p. 1255)	
PRL_ERR_DISK_READ_OUT_DISK (p. 1255)	
PRL_ERR_DISK_RENAME_ERROR (p. 1256)	
PRL_ERR_DISK_RESERVED_WORD (p. 1256)	
PRL_ERR_DISK_RESIZE_FAILED (p. 1256)	
PRL_ERR_DISK_RESIZE_SIZE_TOO_LOW (p. 1256)	
PRL_ERR_DISK_RESIZE_WITH_SNAPSHOTS_NOT_ALLOWED (p. 1257)	
PRL_ERR_DISK_RESIZER_NOT_FOUND (p. 1257)	VM image resizer errors +++ ALIGNED VALUE TO 41000
PRL_ERR_DISK_SET_CACHING_FAILED (p. 1257)	
PRL_ERR_DISK_SET_FILE_SIZE_FAILED (p. 1257)	
PRL_ERR_DISK_SHARED_BLOCK (p. 1258)	• Compressing errors * +++ ALIGNED VALUE TO 22000
PRL_ERR_DISK_SHARING_VIOLATION (p. 1258)	
PRL_ERR_DISK_SMALL_IMAGE_SIZE (p. 1258)	
PRL_ERR_DISK_SNAPSHOTS_CORRUPTED (p. 1258)	
PRL_ERR_DISK_STATES_ERROR (p. 1258)	
PRL_ERR_DISK_STORAGE_CORRUPTED (p. 1259)	
PRL_ERR_DISK_TOOL_NOT_FOUND (p. 1259)	
PRL_ERR_DISK_TOOL_PROCESS_ERROR (p. 1259)	
PRL_ERR_DISK_TRUNCATE_FAILED (p. 1259)	
PRL_ERR_DISK_UNALIGNED (p. 1260)	
PRL_ERR_DISK_UNCOMMITED_OPERATION (p. 1260)	
PRL_ERR_DISK_UNMOUNT_FAILED (p. 1260)	
PRL_ERR_DISK_USER_INTERRUPTED (p. 1260)	
PRL_ERR_DISK_WRITE_FAILED (p. 1260)	
PRL_ERR_DISK_WRITE_OUT_DISK (p. 1261)	
PRL_ERR_DISK_WRITE_REAL_FAILED (p. 1261)	
PRL_ERR_DISK_XML_DIFFERS_FROM_REAL (p. 1261)	
PRL_ERR_DISK_XML_INVALID (p. 1261)	
PRL_ERR_DISK_XML_INVALID_VERSION (p. 1261)	
PRL_ERR_DISK_XML_LARGE_FILE (p. 1262)	
PRL_ERR_DISK_XML_LOCKED (p. 1262)	
PRL_ERR_DISK_XML_MISSING (p. 1262)	
PRL_ERR_DISK_XML_OPEN_FAILED (p. 1262)	
PRL_ERR_DISK_XML_PARTITION_NOT_FOUND (p. 1262)	
PRL_ERR_DISK_XML_SAVE_ERROR (p. 1263)	
PRL_ERR_DISK_XML_SAVE_REMOVE_ERROR (p. 1263)	

PRL_ERR_DISK_XML_SAVE_RENAME_ERROR (p. 1263)	
PRL_ERR_DISP_AUTO_COMPRESS_PERIOD_OUT_OF_RANGE (p. 1263)	
PRL_ERR_DISP_CANNOT_COMPACT_VM_HARD_DISK (p. 1264)	
PRL_ERR_DISP_CONFIG_ALREADY_EXISTS (p. 1264)	
PRL_ERR_DISP_CONFIG_FILE_NOT_SET (p. 1264)	
PRL_ERR_DISP_CONFIG_WRITE_ERR (p. 1264)	
PRL_ERR_DISP_LOGON_ACTIONS_REACHED_UP_LIMIT (p. 1265)	
PRL_ERR_DISP_SHUTDOWN_IN_PROCESS (p. 1265)	
PRL_ERR_DISP_START_VM_FOR_COMPACT_FAILED (p. 1265)	
PRL_ERR_DISP_TIME_MACHINE_IS_RUNNING (p. 1265)	
PRL_ERR_DISP_VM_COMMAND_CANT_BE_EXECUTED (p. 1266)	
PRL_ERR_DISP_VM_IS_NOT_STARTED (p. 1266)	Dispatcher errors codes
PRL_ERR_DISP_VM_IS_NOT_STOPPED (p. 1266)	
PRL_ERR_DISP2DISP_SESSION_ALREADY_AUTHORIZED (p. 1266)	Dispatcher-dispatcher protocol errors codes +++ ALIGNED VALUE TO 30000
PRL_ERR_DISP2DISP_WRONG_USER_SESSION_UUID (p. 1267)	
PRL_ERR_DISPATCHER_SERVICE_MODE (p. 1267)	
PRL_ERR_DISPLAY_ENCODINGS_NOT_SUPPORTED (p. 1267)	
PRL_ERR_DISPLAY_FORMAT_NOT_SUPPORTED (p. 1267)	
PRL_ERR_DOUBLE_INIT (p. 1267)	
PRL_ERR_DROP_SUSPEND_FOR_HDD_QUEST (p. 1268)	
PRL_ERR_ENCRYPTION_COMMIT_PROHIBITED (p. 1268)	
PRL_ERR_ENCRYPTION_COMMIT_REJECTED (p. 1268)	Disk encryption errors +++ ALIGNED VALUE TO 54200
PRL_ERR_ENTRY_ALREADY_EXISTS (p. 1268)	
PRL_ERR_ENTRY_DIR_ALREADY_EXISTS (p. 1269)	
PRL_ERR_ENTRY_DIR_DOES_NOT_EXIST (p. 1269)	
PRL_ERR_ENTRY_DOES_NOT_EXIST (p. 1269)	
PRL_ERR_EXCEED_LIMIT_MAX_RUNNING_VMS (p. 1269)	
PRL_ERR_EXCEED_MEMORY_LIMIT (p. 1269)	
PRL_ERR_EXT_DISK_ALREADY_EXISTS (p. 1270)	
PRL_ERR_EXT_DISK_CANT_RENAME (p. 1270)	
PRL_ERR_FAILED_TO_AUTH_ON_BACKUP_SERVER (p. 1270)	
PRL_ERR_FAILED_TO_CONNECT_TO_BACKUP_SERVER (p. 1270)	
PRL_ERR_FAILED_TO_START_VNC_SERVER (p. 1270)	
PRL_ERR_FAILED_TO_STOP_VNC_SERVER (p. 1271)	
PRL_ERR_FAILURE (p. 1271)	
PRL_ERR_FAILURE_ON_VM_DESTROYED (p. 1271)	
PRL_ERR_FDD_IMAGE_CLONE_TO_SELF (p. 1271)	
PRL_ERR_FDD_IMAGE_COPY (p. 1272)	
PRL_ERR_FDD_IMAGE_NOT_SPECIFIED (p. 1272)	
PRL_ERR_FILE_DISK_SPACE_ERROR (p. 1272)	
PRL_ERR_FILE_NOT_EXIST (p. 1272)	
PRL_ERR_FILE_NOT_FOUND (p. 1272)	
PRL_ERR_FILE_OR_DIR_ALREADY_EXISTS (p. 1273)	

PRL_ERR_FILE_READ_ERROR (p. 1273)	
PRL_ERR_FILE_TRANSFER_CANT_CREATE_DST_FILE (p. 1273)	
PRL_ERR_FILE_TRANSFER_CANT_LOCATE_SRC_FILE (p. 1273)	
PRL_ERR_FILE_TRANSFER_CANT_READ_SRC_FILE (p. 1274)	
PRL_ERR_FILE_TRANSFER_CANT_WRITE_DATA_TO_DST_FILE (p. 1274)	
PRL_ERR_FILE_TRANSFER_CLIENT_NOT_CONNECTED (p. 1274)	• File Transfer errors +++ ALIGNED VALUE TO 26000
PRL_ERR_FILE_TRANSFER_DST_FILE_ALREADY_EXIST (p. 1274)	
PRL_ERR_FILE_TRANSFER_INVALID_ARGUMENTS (p. 1275)	
PRL_ERR_FILE_TRANSFER_INVALID_CREDENTIALS (p. 1275)	
PRL_ERR_FILE_TRANSFER_OPERATION_NOT_SUPPORTED (p. 1275)	
PRL_ERR_FILE_TRANSFER_UPLOAD_CANCELED_BY_USER (p. 1275)	
PRL_ERR_FILE_WRITE_ERROR (p. 1276)	
PRL_ERR_FILECOPY_CANT_CREATE_DIR (p. 1276)	
PRL_ERR_FILECOPY_CANT_OPEN_FILE (p. 1276)	
PRL_ERR_FILECOPY_CANT_WRITE (p. 1276)	
PRL_ERR_FILECOPY_DIR_EXIST (p. 1276)	
PRL_ERR_FILECOPY_FILE_EXIST (p. 1277)	
PRL_ERR_FILECOPY_INTERNAL (p. 1277)	
PRL_ERR_FILECOPY_PROTOCOL (p. 1277)	
PRL_ERR_FIREWALL_TOOL_EXECUTED_WITH_ERROR (p. 1277)	
PRL_ERR_FIXME (p. 1277)	
PRL_ERR_FLOPPY_DRIVE_INVALID (p. 1278)	
PRL_ERR_FLOPPY_IMAGE_ALREADY_EXIST (p. 1278)	
PRL_ERR_FLOPPY_IMAGE_NOT_EXIST (p. 1278)	
PRL_ERR_FORCE_REG_ON_SHARED_STORAGE_IS_NEEDED (p. 1278)	
PRL_ERR_FREE_DISC_SPACE_FOR_CLONE (p. 1279)	
PRL_ERR_FREE_DISK_SPACE_FOR_CREATE_SNAPSHOT (p. 1279)	
PRL_ERR_FREE_DISK_SPACE_FOR_REVERT_TO_SNAPSHOT (p. 1279)	
PRL_ERR_GET_DISK_FREE_SPACE_FAILED (p. 1279)	
PRL_ERR_GET_LICENSE_INVALID_ARG (p. 1280)	
PRL_ERR_GET_MON_STATE_INVALID_ARG (p. 1280)	
PRL_ERR_GET_MON_STATE_VM_NOT_CONFIGURED (p. 1280)	
PRL_ERR_GET_MON_STATE_VM_NOT_CREATED (p. 1280)	
PRL_ERR_GET_NET_SERVICE_STATUS_FAILED (p. 1280)	
PRL_ERR_GET_RESOLUTION_TOOL_INVALID_ARG (p. 1281)	
PRL_ERR_GET_RESOLUTION_TOOL_VMNOTCREATED (p. 1281)	
PRL_ERR_GET_USER_HOME_DIR (p. 1281)	
PRL_ERR_GUEST_MAC_INVALID_VERSION (p. 1281)	
PRL_ERR_GUEST_MAC_NOT_ENOUGH_MEMORY (p. 1281)	
PRL_ERR_GUEST_MAC_NOT_MACSERVER_HOST (p. 1282)	
PRL_ERR_GUEST_PROGRAM_EXECUTION_FAILED (p. 1282)	
PRL_ERR_GUEST_TOOLS_NOT_INSTALLED (p. 1282)	
PRL_ERR_HANDSHAKE_FAILED (p. 1282)	

PRL_ERR_HARD_DISK_IMAGE_CORRUPTED (p. 1283)	
PRL_ERR_HARD_DISK_NOT_VIRTUAL_OR_DISABLED (p. 1283)	
PRL_ERR_HDD_IMAGE_CLONE_TO_SELF (p. 1283)	
PRL_ERR_HDD_IMAGE_COPY (p. 1283)	
PRL_ERR_HDD_IMAGE_IS_ALREADY_EXIST (p. 1283)	
PRL_ERR_HDD_IMAGE_NOT_EXIST (p. 1284)	
PRL_ERR_HDD_IMAGE_NOT_SPECIFIED (p. 1284)	
PRL_ERR_HOST_AMD_GUEST_MAC (p. 1284)	
PRL_ERR_HTTP_AUTH_REQUIRED (p. 1284)	
PRL_ERR_HTTP_CONNECTION_REFUSED (p. 1285)	
PRL_ERR_HTTP_HOST_NOT_FOUND (p. 1285)	Problem report send errors set +++ ALIGNED VALUE TO 38000
PRL_ERR_HTTP_INVALID_RESPONSE_HEADER (p. 1285)	
PRL_ERR_HTTP_PROBLEM_REPORT_SEND_FAILURE (p. 1285)	
PRL_ERR_HTTP_PROXY_AUTH_REQUIRED (p. 1286)	
PRL_ERR_HTTP_REQUEST_FAILED (p. 1286)	
PRL_ERR_HTTP_UNEXPECTED_CLOSE (p. 1286)	
PRL_ERR_HTTP_WRONG_CONTENT_LENGTH (p. 1286)	
PRL_ERR_HVT_DISABLED (p. 1286)	
PRL_ERR_HVT_DISABLED_WARNING (p. 1287)	
PRL_ERR_HVT_NOT_AVAILABLE_WARNING (p. 1287)	
PRL_ERR_HVT_NOT_PRESENT (p. 1287)	
PRL_ERR_HVT_NOT_PRESENT_WARNING (p. 1287)	
PRL_ERR_HVT_TURNED_OFF_IN_CONFIG (p. 1287)	
PRL_ERR_HYP_ALLOC_VMS_MEMORY (p. 1288)	
PRL_ERR_IMAGE_NEED_TO_CONVERT (p. 1288)	
PRL_ERR_IMPERSONATE_FAILED (p. 1288)	
PRL_ERR_IMPORT_BOOTCAMP_VM_FAILED (p. 1288)	
PRL_ERR_IMPORT_BOOTCAMP_VM_NO_SPACE (p. 1288)	
PRL_ERR_INCONSISTENCY_VM_CONFIG (p. 1289)	
PRL_ERR_INCORRECT_CDROM_PATH (p. 1289)	
PRL_ERR_INCORRECT_FDD_PATH (p. 1289)	
PRL_ERR_INCORRECT_PATH (p. 1289)	
PRL_ERR_INSERTING_VM_TO_CATALOGUE (p. 1289)	
PRL_ERR_INSTALLATION_PROBLEM (p. 1290)	
PRL_ERR_INVALID_ACCESS_TOKEN_RECEIVED (p. 1290)	
PRL_ERR_INVALID_ACTION_REQUESTED (p. 1290)	
PRL_ERR_INVALID_ARG (p. 1290)	
PRL_ERR_INVALID_CONFIGURATION (p. 1290)	
PRL_ERR_INVALID_CREATE_FLOPPY_IMAGE_PARAMETERS (p. 1291)	
PRL_ERR_INVALID_HANDLE (p. 1291)	
PRL_ERR_INVALID_HDD_GEOMETRY (p. 1291)	
PRL_ERR_INVALID_KEXT_REBOOT_REQUIRED (p. 1291)	
PRL_ERR_INVALID_MEMORY_GUARANTEE (p. 1292)	
PRL_ERR_INVALID_MEMORY_QUOTA (p. 1292)	

PRL_ERR_INVALID_MEMORY_SIZE (p. 1292)	
PRL_ERR_INVALID_OS_TYPE (p. 1292)	
PRL_ERR_INVALID_PARALLELS_DISK (p. 1292)	• Hard Drive +++ ALIGNED VALUE TO 2000
PRL_ERR_INVALID_PARAM (p. 1293)	
PRL_ERR_INVALID_TOTAL_LIMIT (p. 1293)	
PRL_ERR_IO_AUTHENTICATION_FAILED (p. 1293)	
PRL_ERR_IO_CONNECTION_TIMEOUT (p. 1293)	
PRL_ERR_IO_DISABLED (p. 1294)	
PRL_ERR_IO_INVALID_POINTER_ACCESS (p. 1294)	
PRL_ERR_IO_NO_CONNECTION (p. 1294)	
PRL_ERR_IO_SEND_QUEUE_IS_FULL (p. 1294)	
PRL_ERR_IO_STOPPED (p. 1294)	
PRL_ERR_IO_UNKNOWN_VM_ID (p. 1295)	
PRL_ERR_IOSERVICE_COMPRESS (p. 1295)	
PRL_ERR_IPC_ATTACH_FAILED (p. 1295)	
PRL_ERR_IPC_CREATE_MAP_FAILED (p. 1295)	
PRL_ERR_IPC_FTOK_FAILED (p. 1295)	
PRL_ERR_IPFO_INVALID_MODE (p. 1296)	
PRL_ERR_IPFO_RECEIVE_FAILED (p. 1296)	
PRL_ERR_IPFO_SEND_FAILED (p. 1296)	
PRL_ERR_IPFO_SOCKET_ACCEPT_FAILED (p. 1296)	
PRL_ERR_IPFO_SOCKET_BIND_FAILED (p. 1296)	
PRL_ERR_IPFO_SOCKET_CONNECT_FAILED (p. 1297)	
PRL_ERR_IPFO_SOCKET_CREATE_FAILED (p. 1297)	• Errors for IPC file open library +++ ALIGNED VALUE TO 29000
PRL_ERR_IPFO_SOCKET_LISTEN_FAILED (p. 1297)	
PRL_ERR_IPFO_SOCKET_NOT_OPENED (p. 1297)	
PRL_ERR_IPHONE_PROXY_ALREADY_STARTED (p. 1297)	
PRL_ERR_IPHONE_PROXY_CANNOT_START (p. 1298)	iPhone proxy errors +++ ALIGNED VALUE TO 32000
PRL_ERR_IPHONE_PROXY_CANNOT_STOP (p. 1298)	
PRL_ERR_ISCSI_STORAGE_ALREADY_REGISTERED (p. 1298)	
PRL_ERR_ISCSI_STORAGE_CANNOT_CREATE_MOUNT_POINT (p. 1298)	
PRL_ERR_ISCSI_STORAGE_CREATE (p. 1299)	
PRL_ERR_ISCSI_STORAGE_EXTEND (p. 1299)	
PRL_ERR_ISCSI_STORAGE_GET_STATE (p. 1299)	
PRL_ERR_ISCSI_STORAGE_INVALID_FSTYPE (p. 1299)	
PRL_ERR_ISCSI_STORAGE_LIBRARY (p. 1299)	iscsi storages errors +++ ALIGNED VALUE TO 44400
PRL_ERR_ISCSI_STORAGE_MOUNT (p. 1300)	
PRL_ERR_ISCSI_STORAGE_MOUNT_POINT_ALREADY_EXISTS (p. 1300)	

PRL_ERR_ISCSI_STORAGE_MOUNTED (p. 1300)	
PRL_ERR_ISCSI_STORAGE_NOT_FOUND (p. 1300)	
PRL_ERR_ISCSI_STORAGE_NOT_MOUNTED (p. 1301)	
PRL_ERR_ISCSI_STORAGE_NOT_SUPPORTED (p. 1301)	
PRL_ERR_ISCSI_STORAGE_REMOVE (p. 1301)	
PRL_ERR_ISCSI_STORAGE_START (p. 1301)	
PRL_ERR_ISCSI_STORAGE_UMOUNT (p. 1301)	
PRL_ERR_IT_CANT_COMPACT (p. 1302)	compact
PRL_ERR_IT_CANT_COMPACT_LDM_DISK (p. 1302)	
PRL_ERR_IT_CANT_COMPACT_PLAIN_DISK (p. 1302)	
PRL_ERR_IT_CANT_GET_BITMAP (p. 1302)	
PRL_ERR_IT_CANT_GET_PARAMS_FROM_OLD_IMAGE (p. 1303)	
PRL_ERR_IT_CANT_OPEN_IMAGE (p. 1303)	
PRL_ERR_IT_CANT_PREPARE_RECONFIG_DATA (p. 1303)	
PRL_ERR_IT_CANT_RECONFIG (p. 1303)	
PRL_ERR_IT_CANT_RECONFIG_BOOTCAMP (p. 1303)	
PRL_ERR_IT_CANT_RESIZE_LDM_DISK (p. 1304)	
PRL_ERR_IT_CANT_RESIZE_VOLUME (p. 1304)	resize
PRL_ERR_IT_CANT_VALIDATE_BOOTCAMP (p. 1304)	reconfig
PRL_ERR_IT_CONVERT_TO_CURRENT_ERROR (p. 1304)	convert
PRL_ERR_IT_DISK_SMALL_FOR_SPLIT (p. 1305)	
PRL_ERR_IT_FAST_RESIZE_FAILURE (p. 1305)	
PRL_ERR_IT_FIRST_ERROR (p. 1305)	ALIGNED VALUE TO 46200
PRL_ERR_IT_FIRST_FAILURE (p. 1305)	ALIGNED VALUE TO 46100
PRL_ERR_IT_FS_NOT_SUPPORTED_FOR_RESIZE (p. 1305)	
PRL_ERR_IT_GUEST_TOOLS_UNKNOWN_STATE (p. 1306)	guest tools
PRL_ERR_IT_HAL_NOT_FOUND (p. 1306)	
PRL_ERR_IT_RECONFIG_PATH_NEEDED (p. 1306)	
PRL_ERR_IT_ROLLBACK_FAILURE (p. 1306)	
PRL_ERR_IT_SIZE_CHANGED (p. 1307)	
PRL_ERR_IT_SNAPSHOTS_MERGE_IS_NEEDED (p. 1307)	merge
PRL_ERR_IT_UNKNOWN_OS (p. 1307)	
PRL_ERR_IT_USER_INTERRUPTED (p. 1307)	
PRL_ERR_LIC_REGISTRATION_COMMON_ERROR (p. 1307)	
PRL_ERR_LICENSE_AUTH_FAILED (p. 1308)	
PRL_ERR_LICENSE_BETA_KEY_RELEASE_PRODUCT (p. 1308)	
PRL_ERR_LICENSE_BLACKLISTED (p. 1308)	
PRL_ERR_LICENSE_BLACKLISTED_TO_VM_OPERATION (p. 1308)	
PRL_ERR_LICENSE_DEFERRED_LICENSE_NOT_FOUND (p. 1309)	
PRL_ERR_LICENSE_EXPIRED (p. 1309)	
PRL_ERR_LICENSE_FILE_WRITE_FAILED (p. 1309)	
PRL_ERR_LICENSE_GRACED (p. 1309)	
PRL_ERR_LICENSE_IS_NOT_VOLUME (p. 1309)	
PRL_ERR_LICENSE_NOT_STARTED (p. 1310)	
PRL_ERR_LICENSE_NOT_VALID (p. 1310)	

PRL_ERR_LICENSE_RELEASE_KEY_BETA_PRODUCT (p. 1310)	
PRL_ERR_LICENSE_RESTRICTED_CPU_COUNT (p. 1310)	
PRL_ERR_LICENSE_RESTRICTED_GUEST_OS (p. 1311)	
PRL_ERR_LICENSE_RESTRICTED_TO_CLONE_VM (p. 1311)	
PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_FROM_TEMPLATE (p. 1311)	
PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_TO_TEMPLATE (p. 1311)	
PRL_ERR_LICENSE_RESTRICTED_TO_CREATE_VM (p. 1312)	
PRL_ERR_LICENSE_RESTRICTED_TO_PAUSE_VM (p. 1312)	
PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_3RD_PARTY_VM (p. 1312)	
PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_VM (p. 1312)	
PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT (p. 1313)	
PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT_PER_USER (p. 1313)	
PRL_ERR_LICENSE_RESTRICTED_TO_SAFEMODE_FEATURE (p. 1313)	
PRL_ERR_LICENSE_RESTRICTED_TO_SMARTGUARD_FEATURE (p. 1314)	
PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_CREATE (p. 1314)	
PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_DELETE (p. 1314)	
PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SHOW_TREE (p. 1315)	
PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SWITCH (p. 1315)	
PRL_ERR_LICENSE_RESTRICTED_TO_SUSPEND_VM (p. 1315)	
PRL_ERR_LICENSE_RESTRICTED_TO_UNDODISK_FEATURE (p. 1315)	
PRL_ERR_LICENSE_TOO_MANY_MEMORY (p. 1316)	
PRL_ERR_LICENSE_TOO_MANY_VCPUS (p. 1316)	
PRL_ERR_LICENSE_UNSUPPORTED_LICENSE_TYPE_TO_DEACTIVATION (p. 1316)	
PRL_ERR_LICENSE_UPGRADE_NO_ACCEPTABLE_LICENSE (p. 1316)	
PRL_ERR_LICENSE_UPGRADE_NO_PERMANENT_LICENSE (p. 1317)	
PRL_ERR_LICENSE_VALID (p. 1317)	• License +++ ALIGNED VALUE TO 11000
PRL_ERR_LICENSE_VM_HAS_VTD_DEVICES (p. 1317)	
PRL_ERR_LICENSE_WRONG_ADVANCED_FIELD (p. 1317)	
PRL_ERR_LICENSE_WRONG_DISTRIBUTOR (p. 1318)	
PRL_ERR_LICENSE_WRONG_LANGUAGE (p. 1318)	
PRL_ERR_LICENSE_WRONG_PLATFORM (p. 1318)	
PRL_ERR_LICENSE_WRONG_PRODUCT (p. 1318)	
PRL_ERR_LICENSE_WRONG_VERSION (p. 1318)	
PRL_ERR_LOCAL_AUTHENTICATION_FAILED (p. 1319)	
PRL_ERR_LOGIN_BY_MISMATCH_CLIENT_MODE (p. 1319)	
PRL_ERR_LOW_MEMORY_LIMIT (p. 1319)	
PRL_ERR_LPT_DEVICE_ALREADY_EXIST (p. 1319)	
PRL_ERR_LPT_OUTPUT_FILE_IS_NOT_SPECIFIED (p. 1319)	
PRL_ERR_LPT_OUTPUT_FILE_NOT_EXIST (p. 1320)	
PRL_ERR_MAC_ADDRESS_IN_CORRECT_LENGTH (p. 1320)	
PRL_ERR_MAC_ADDRESS_INCORRECT (p. 1320)	
PRL_ERR_MAC_ADDRESS_IS_EMPTY (p. 1320)	
PRL_ERR_MAC_ADDRESS_WITH_2_ZERO_START (p. 1320)	

PRL_ERR_MAC_ADDRESS_WITH_ALL_ZEROS (p. 1321)	
PRL_ERR_MAC_ADDRESS_WITHIN_CORRECT_SYMBOLS (p. 1321)	
PRL_ERR_MAKE_DIRECTORY (p. 1321)	
PRL_ERR_MEM_EXCEED_PHY_SPACE (p. 1321)	
PRL_ERR_MEMORY_ALLOC_ERROR (p. 1321)	
PRL_ERR_MERGE_XML_DOCUMENT_CONFLICT (p. 1322)	
PRL_ERR_MOBILE_ADVANCED_AUTH_REQUIRED (p. 1322)	
PRL_ERR_MOUNTER_LIST_NO_OBJECT (p. 1322)	
PRL_ERR_MSG_START_32BIT_VM_ON_64BIT_HOST (p. 1322)	
PRL_ERR_NEED_KILL_PREV_SETTING (p. 1323)	
PRL_ERR_NETWORK_ADAPTER_NOT_FOUND (p. 1323)	
PRL_ERR_NETWORK_ROLLBACK_FAILED (p. 1323)	
PRL_ERR_NO_BOOTING_DEVICE_SELECTED (p. 1323)	
PRL_ERR_NO_CD_DRIVE_AVAILABLE (p. 1323)	
PRL_ERR_NO_DATA (p. 1324)	
PRL_ERR_NO_DISP_CONFIG_FOUND (p. 1324)	
PRL_ERR_NO_GUEST_OS_FOUND (p. 1324)	
PRL_ERR_NO_MORE_FREE_INTERFACE_SLOTS (p. 1324)	
PRL_ERR_NO_ONE_HARD_DISK_TO_COMPRESS (p. 1324)	
PRL_ERR_NO_PROBLEM_REPORT_FOUND (p. 1325)	
PRL_ERR_NO_TARGET_DIR_PATH_SPECIFIED (p. 1325)	
PRL_ERR_NO_TARGET_PATH_SPECIFIED (p. 1325)	
PRL_ERR_NO_VM_DIR_CONFIG_FOUND (p. 1325)	
PRL_ERR_NOT_ALL_FILES_WAS_DELETED (p. 1325)	
PRL_ERR_NOT_CONNECTED_TO_DISPATCHER (p. 1326)	
PRL_ERR_NOT_CONNECTED_TO_PROXY_MANAGER (p. 1326)	
PRL_ERR_NOT_ENOUGH_DISK_FREE_SPACE (p. 1326)	
PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_START_VM (p. 1326)	
PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_XML_SAVE (p. 1327)	
PRL_ERR_NOT_ENOUGH_PERMS_TO_OPEN_AUTHORIZATION_FILE (p. 1327)	
PRL_ERR_NOT_ENOUGH_RIGHTS_FOR_LINKED_CLONE (p. 1327)	
PRL_ERR_NOT_LOCK_OWNER_SESSION_TRIES_TO_UNLOCK (p. 1327)	
PRL_ERR_NOT_SENTILLION_CLIENT (p. 1328)	ALIGNED VALUE TO 39000
PRL_ERR_NVRAM_FILE_COPY (p. 1328)	
PRL_ERR_OBJECT_BAD_INTERFACE (p. 1328)	
PRL_ERR_OBJECT_CLASS_NOT_FOUND (p. 1328)	
PRL_ERR_OBJECT_DUPLICATE_CLASS (p. 1328)	
PRL_ERR_OBJECT_DUPLICATE_UID (p. 1329)	
PRL_ERR_OBJECT_LIB_CANT_GET_PERMS (p. 1329)	
PRL_ERR_OBJECT_LIB_LOAD_ERROR (p. 1329)	
PRL_ERR_OBJECT_LIB_NO_FUNCTIONS (p. 1329)	
PRL_ERR_OBJECT_LIB_WRONG_PERMS (p. 1330)	

PRL_ERR_OBJECT_NOT_FOUND (p. 1330)	• Shared objects manipulation errors +++ ALIGNED VALUE TO 42000
PRL_ERR_OBJECT_WAS_REMOVED (p. 1330)	
PRL_ERR_ONLY_ADMIN_CAN_SET_PARAMETER_STARTLOGINMODE_ROOT (p. 1330)	
PRL_ERR_ONLY_ADMIN_CAN_SET_VERBOSE_LOGGING (p. 1331)	
PRL_ERR_ONLY_ADMIN_OR_VM_OWNER_CAN_OPEN_THIS_SESSION (p. 1331)	VM guest operations errors +++ ALIGNED VALUE TO 34000
PRL_ERR_OPEN_DISP_CONFIG_READ (p. 1331)	
PRL_ERR_OPEN_DISP_CONFIG_WRITE (p. 1331)	
PRL_ERR_OPEN_FAILED (p. 1332)	
PRL_ERR_OPEN_PROBLEM_REPORT_READ (p. 1332)	
PRL_ERR_OPEN_VM_CONFIG_READ (p. 1332)	
PRL_ERR_OPEN_VM_CONFIG_WRITE (p. 1332)	
PRL_ERR_OPEN_VM_DIR_CONFIG_READ (p. 1332)	
PRL_ERR_OPEN_VM_DIR_CONFIG_WRITE (p. 1333)	
PRL_ERR_OPERATION_FAILED (p. 1333)	
PRL_ERR_OPERATION_PENDING (p. 1333)	
PRL_ERR_OPERATION_WAS_CANCELED (p. 1333)	
PRL_ERR_OS_RECONFIG_DATA_ABSENT (p. 1333)	
PRL_ERR_OUT_OF_DISK_SPACE (p. 1334)	
PRL_ERR_OUT_OF_MEMORY (p. 1334)	
PRL_ERR_PARALLEL_PORT_IMAGE_NOT_EXIST (p. 1334)	
PRL_ERR_PARALLEL_PORT_IMG_COPY (p. 1334)	
PRL_ERR_PARALLEL_PORT_NO_RASTERIZER (p. 1334)	
PRL_ERR_PARAM_NOT_FOUND (p. 1335)	
PRL_ERR_PARSE_CLIENT_PREFS (p. 1335)	
PRL_ERR_PARSE_COMMON_SERVER_PREFS (p. 1335)	
PRL_ERR_PARSE_DISP_CONFIG (p. 1335)	
PRL_ERR_PARSE_FILESYSTEM_INFO (p. 1335)	
PRL_ERR_PARSE_HARD_DISK_HW_INFO (p. 1336)	
PRL_ERR_PARSE_HOST_HW_INFO (p. 1336)	
PRL_ERR_PARSE_PROBLEM_REPORT (p. 1336)	
PRL_ERR_PARSE_STATISTICS (p. 1336)	
PRL_ERR_PARSE_USER_PROFILE (p. 1336)	
PRL_ERR_PARSE_VM_CONFIG (p. 1337)	
PRL_ERR_PARSE_VM_DIR_CONFIG (p. 1337)	
PRL_ERR_PARSING_EVENT (p. 1337)	
PRL_ERR_PATH_IS_NOT_DIRECTORY (p. 1337)	
PRL_ERR_PAX_ACCESS_FORBIDDEN (p. 1337)	
PRL_ERR_PAX_HOST_LIMIT_WAS_EXCEEDED (p. 1338)	
PRL_ERR_PCMOVER_EXEC_FAILED (p. 1338)	
PRL_ERR_PCMOVER_MIGRATE_FAILED (p. 1338)	

PRL_ERR_PCMOVER_NOT_INSTALLED (p. 1338)	PCmover managment library errors +++ ALIGNED VALUE TO 43100
PRL_ERR_PCMOVER_POLICY_FILE_OPEN_ERROR (p. 1338)	
PRL_ERR_PCMOVER_POLICY_FILE_WRITE_ERROR (p. 1339)	
PRL_ERR_PCMOVER_VAN_FILE_CREATE_ERROR (p. 1339)	
PRL_ERR_PCMOVER_VAN_FILE_OPEN_ERROR (p. 1339)	
PRL_ERR_PCMOVER_VAN_FILE_PREPARE_FAILED (p. 1339)	
PRL_ERR_PCMOVER_WINDOWS_DIR_NOT_EXIST (p. 1340)	
PRL_ERR_PERFORM_MOUNTER_COMMAND (p. 1340)	
PRL_ERR_PLAYER_CANT_SUSPEND_IN_PAUSE (p. 1340)	
PRL_ERR_PPC_INVALID_CREDENTIALS (p. 1340)	• Proxy connection errors +++ ALIGNED VALUE TO 600
PRL_ERR_PPC_INVALID_CREDENTIALS_ON_RECONNECT (p. 1341)	
PRL_ERR_PPC_SERVER_BUSY (p. 1341)	
PRL_ERR_PPC_UNABLE_TO_CONNECT (p. 1341)	
PRL_ERR_PREPARE_FOR_HIBERNATE_TASK_ALREADY_RUN (p. 1341)	
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_CANNOT_STAND_BY (p. 1342)	
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WITHOUT_TOOLS (p. 1342)	
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WRONG_STATE (p. 1342)	
PRL_ERR_PREV_SESSION_IS_ACTIVE (p. 1342)	
PRL_ERR_PROBLEM_REPORT_ALREADY_EXISTS (p. 1343)	
PRL_ERR_PROBLEM_REPORT_FILE_NOTSET (p. 1343)	
PRL_ERR_PROBLEM_REPORT_WRITE (p. 1343)	
PRL_ERR_PROXY_HANDSHAKE_FAILED (p. 1343)	
PRL_ERR_PROXY_PEER_NOT_FOUND (p. 1343)	
PRL_ERR_PROXY_WRONG_PORT_NUMBER (p. 1344)	
PRL_ERR_PROXY_WRONG_PROTOCOL_VERSION (p. 1344)	
PRL_ERR_READ_FAILED (p. 1344)	
PRL_ERR_READ_XML_CONTENT (p. 1344)	
PRL_ERR_READONLY_FILESYSTEM (p. 1344)	
PRL_ERR_REBOOT_HOST (p. 1345)	
PRL_ERR_REG_PSTORAGE_REVOKE_IS_NEEDS (p. 1345)	
PRL_ERR_REMOTE_DEVICE_EXIST (p. 1345)	
PRL_ERR_REMOTE_DEVICE_NOT_EXIST (p. 1345)	
PRL_ERR_REMOTE_DEVICE_NOT_REGISTERED (p. 1345)	Device of this type cannot be remote device type is not registered
PRL_ERR_REMOTE_DISPLAY_EMPTY_PASSWORD (p. 1346)	
PRL_ERR_REMOTE_DISPLAY_HOST_NOT_SPECIFIED (p. 1346)	
PRL_ERR_REMOTE_DISPLAY_WRONG_PORT_NUMBER (p. 1346)	
PRL_ERR_RESUME_BOOTCAMP_CHANGED_DISK_CONTENTS (p. 1346)	
PRL_ERR_RESUME_BOOTCAMP_CORRUPT_DISK_STATE_PARAM (p. 1347)	
PRL_ERR_RETRIEVE_VM_CONFIG (p. 1347)	
PRL_ERR_RETURN_CODE_RANG_EEND (p. 1347)	

PRL_ERR_REVERT_IMPERSONATE_FAILED (p. 1347)	
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT (p. 1348)	
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_PAUSED (p. 1348)	
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL (p. 1348)	
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL (p. 1348)	
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS (p. 1349)	
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL (p. 1349)	
PRL_ERR_RUN_VM_ACTION_SCRIPT (p. 1349)	
PRL_ERR_RUNNING_VM_OR_CT (p. 1349)	
PRL_ERR_SAFE_MODE_START_DURING_SUSPENDING_SYNC (p. 1350)	
PRL_ERR_SAMPLE_CONFIG_NOT_FOUND (p. 1350)	
PRL_ERR_SAVE_VM_CATALOG (p. 1350)	
PRL_ERR_SAVE_VM_CONFIG (p. 1350)	
PRL_ERR_SDK_TRY_AGAIN (p. 1351)	SDK errors +++ ALIGNED VALUE TO 50000
PRL_ERR_SEARCH_CONFIG_OPERATION_CANCELED (p. 1351)	
PRL_ERR_SEND_COMMAND_TOWS_FAILED (p. 1351)	
PRL_ERR_SERIAL_IMG_COPY (p. 1351)	
PRL_ERR_SERIAL_IMG_NOT_FOUND (p. 1352)	
PRL_ERR_SERIAL_PORT_IMAGE_NOT_EXIST (p. 1352)	
PRL_ERR_SERVER_GOES_DOWN (p. 1352)	
PRL_ERR_SERVER_PREFS_EDIT_COLLISION (p. 1352)	
PRL_ERR_SERVICE_BUSY (p. 1352)	
PRL_ERR_SET_CPULIMIT (p. 1353)	Virtuozzo specific error +++ ALIGNED VALUE TO 44000
PRL_ERR_SET_CPUMASK (p. 1353)	
PRL_ERR_SET_CPUUNITS (p. 1353)	
PRL_ERR_SET_IOLIMIT (p. 1353)	
PRL_ERR_SET_IOPRIO (p. 1353)	
PRL_ERR_SET_LICENSE_INVALID_ARG (p. 1354)	
PRL_ERR_SET_LICENSE_INVALID_KEY (p. 1354)	
PRL_ERR_SET_NETWORK_SETTINGS_FAILED (p. 1354)	
PRL_ERR_SHUT_DOWN_NOT_IFICATION_SERVICE (p. 1354)	
PRL_ERR_SMC_ERROR (p. 1355)	
PRL_ERR_SMP_NOT_SUPPORTED_HVT_DISABLED (p. 1355)	
PRL_ERR_SMP_NOT_SUPPORTED_HVT_NOT_PRESENT (p. 1355)	
PRL_ERR_SNAPSHOTS_COPY (p. 1355)	
PRL_ERR_SOME_TASKS_PRESENT (p. 1355)	
PRL_ERR_SOME_VMS_RUNNING (p. 1356)	
PRL_ERR_SOUND_BAD_EMULATION_TYPE (p. 1356)	
PRL_ERR_SOUND_DEVICE_WRITE_FAILED (p. 1356)	• Sound +++ ALIGNED VALUE TO 5000

PRL_ERR_SOUND_IN_DEVICE_OPEN_FAILED (p. 1356)	
PRL_ERR_SOUND_OUT_DEVICE_OPEN_FAILED (p. 1357)	
PRL_ERR_SOURCE_ISNT_BOOTCAMP_DISK (p. 1357)	
PRL_ERR_SSL_HANDSHAKE_FAILED (p. 1357)	
PRL_ERR_STAND_BY_VM_BY_GUEST_SLEEP_TIMEOUT (p. 1357)	
PRL_ERR_STAND_BY_VM_PAUSED (p. 1357)	
PRL_ERR_STAND_BY_VM_WITH_OUTDATED_SHUTDOWN_TOOL (p. 1358)	
PRL_ERR_STAND_BY_VM_WITH_UNLOADED_SHUTDOWN_TOOL (p. 1358)	
PRL_ERR_STAND_BY_VM_WITH_UNSUPPORTED_SHUTDOWN_TOOL (p. 1358)	
PRL_ERR_START_SAFE_MODE_UNSUPPORTED (p. 1359)	
PRL_ERR_START_VM_BY_NOT_AUTH_USER (p. 1359)	
PRL_ERR_START_VM_BY_NOT_DEFINED_USER (p. 1359)	
PRL_ERR_STATE_ALREADY_EXISTS (p. 1359)	
PRL_ERR_STATE_CANT_LOAD_SPECIFIED_CFG (p. 1359)	
PRL_ERR_STATE_CANT_OPEN_FOR_WRITE (p. 1360)	
PRL_ERR_STATE_CANT_OPEN_IMAGE (p. 1360)	
PRL_ERR_STATE_CANT_OPEN_LOCKED (p. 1360)	
PRL_ERR_STATE_CANT_SAVE_LOCKED (p. 1360)	
PRL_ERR_STATE_DELETE_NONCLOSED (p. 1360)	
PRL_ERR_STATE_ERROR_CREATE_IMAGE (p. 1361)	
PRL_ERR_STATE_ERROR_RENAMING_IMAGE (p. 1361)	
PRL_ERR_STATE_FULL_DELETE_FAILED (p. 1361)	
PRL_ERR_STATE_GETFREESPACE_FAILED (p. 1361)	
PRL_ERR_STATE_INT_CORRUPTED (p. 1361)	
PRL_ERR_STATE_INVALID_IMAGE_TYPE (p. 1362)	
PRL_ERR_STATE_INVALID_PARAMETERS (p. 1362)	
PRL_ERR_STATE_MEMORY_ERROR (p. 1362)	
PRL_ERR_STATE_MERGE_FAILED (p. 1362)	
PRL_ERR_STATE_MERGE_NO_SPACE (p. 1362)	
PRL_ERR_STATE_NO_DISKS (p. 1363)	
PRL_ERR_STATE_NO_STATE (p. 1363)	
PRL_ERR_STATE_NOT_OPENED (p. 1363)	
PRL_ERR_STATE_NOT_PERMITTED (p. 1363)	
PRL_ERR_STATE_NOT_RELEASED (p. 1363)	
PRL_ERR_STATE_PROCESS_RUNNING (p. 1364)	
PRL_ERR_STATE_ROLLBACK_ERROR (p. 1364)	
PRL_ERR_STATE_ROLLBACK_IN_PROGRESS (p. 1364)	
PRL_ERR_STATE_STATFS_FAILED (p. 1364)	
PRL_ERR_STATE_STOPPING_STATE (p. 1364)	
PRL_ERR_STATE_UNEXPECTED_ERROR (p. 1365)	• Range reserved for different states error +++ ALIGNED VALUE TO 19000

PRL_ERR_SUCCESS (p. 1365)	All Virtuozzo error values should go here in order to be maintained in one place We're taking approach of HRESULT windows notation error codes here.
PRL_ERR_SUSPEND_BOOTCAMP_NOT_NTFS_ONLY_DISK (p. 1365)	
PRL_ERR_SUSPEND_BOOTCAMP_NTFS_RW_MOUNTERS_DETECTED (p. 1365)	
PRL_ERR_SUSPEND_REJECTED_BY_GUEST (p. 1366)	
PRL_ERR_SUSPEND_VM_VTD_BY_GUEST_SLEEP_TIMEOUT (p. 1366)	
PRL_ERR_SUSPEND_VM_VTD_PAUSED (p. 1366)	VM error codes +++ ALIGNED VALUE TO 36000
PRL_ERR_SUSPEND_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL (p. 1366)	
PRL_ERR_SUSPEND_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL (p. 1367)	
PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_CAPS (p. 1367)	
PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL (p. 1367)	
PRL_ERR_SUSPEND_WITH_USB_BOOTDISK (p. 1367)	
PRL_ERR_SYMBOL_NOT_FOUND (p. 1368)	
PRL_ERR_TARGET_NAME_ALREADY_OCCUPIED (p. 1368)	
PRL_ERR_TARGET_PATH_IS_NOT_DIRECTORY (p. 1368)	
PRL_ERR_TASK_NOT_FOUND (p. 1368)	
PRL_ERR_TEMPLATE_HAS_APPS (p. 1368)	containers templates errors +++ ALIGNED VALUE TO 46400
PRL_ERR_TEMPLATE_NOT_FOUND (p. 1369)	
PRL_ERR_TEST_TEXT_MESSAGES (p. 1369)	Test errors +++ ALIGNED VALUE TO 43200
PRL_ERR_TEST_TEXT_MESSAGES_PS (p. 1369)	
PRL_ERR_TIME_MACHINE_EXCLUDED_LIST_OP (p. 1369)	
PRL_ERR_TIMEOUT (p. 1370)	
PRL_ERR_TIS_INVALID_UID (p. 1370)	• TIS errors +++ ALIGNED VALUE TO 27800
PRL_ERR_TOO_LOW_HDD_SIZE (p. 1370)	
PRL_ERR_TOOLS_UNSUPPORTED_GUEST (p. 1370)	
PRL_ERR_TRY_AGAIN (p. 1370)	
PRL_ERR_UNABLE_APPLY_MEMORY_GUARANTEE (p. 1371)	
PRL_ERR_UNABLE_APPLY_TOTAL_LIMIT (p. 1371)	
PRL_ERR_UNABLE_DROP_SUSPENDED_STATE (p. 1371)	
PRL_ERR_UNABLE_SEND_REQUEST (p. 1371)	
PRL_ERR_UNABLE_TO_CONTINUE_VM_LIFETIME_IS_EXPIRED (p. 1372)	
PRL_ERR_UNABLE_TO_SEND_REQUEST (p. 1372)	
PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE (p. 1372)	
PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE_BY_NON_PRIVILEGED_USER (p. 1372)	
PRL_ERR_UNATTENDED_UNSUPPORTED_GUEST (p. 1373)	
PRL_ERR_UNDER_OLD_HYPERVISOR (p. 1373)	
PRL_ERR_UNEXPECTED (p. 1373)	
PRL_ERR_UNIMPLEMENTED (p. 1373)	

PRL_ERR_UNINITIALIZED (p. 1373)	
PRL_ERR_UNNAMED_CT_MOVE (p. 1374)	
PRL_ERR_UNRECOGNIZED_REQUEST (p. 1374)	
PRL_ERR_UNSUPPORTED_DEVICE_TYPE (p. 1374)	
PRL_ERR_UNSUPPORTED_FILE_SYSTEM (p. 1374)	
PRL_ERR_UNSUPPORTED_LAYOUTS_STRUCTURE (p. 1374)	
PRL_ERR_UNSUPPORTED_NETWORK_FILE_SYSTEM (p. 1375)	
PRL_ERR_UNSUPPORTED_VIRTUAL_SOURCE (p. 1375)	VI convert errors +++ ALIGNED VALUE TO 45000
PRL_ERR_UPD_TOOLS_VER_VM_NOT_CONFIGURED (p. 1375)	
PRL_ERR_UPD_TOOLS_VER_VM_NOT_CREATED (p. 1375)	
PRL_ERR_UPD_UPDATER_CONFIG (p. 1375)	• Range reserved for updater errors +++ ALIGNED VALUE TO 17000
PRL_ERR_UPD_UPDATES (p. 1376)	
PRL_ERR_UPDATE_MEM_VM_NOT_CREATED (p. 1376)	
PRL_ERR_USER_CANT_CHANGE_ACCESS_PROFILE (p. 1376)	
PRL_ERR_USER_CANT_CHANGE_PROFILE_ACCESS_PART (p. 1376)	
PRL_ERR_USER_CANT_CHANGE_READ_ONLY_VALUE (p. 1377)	
PRL_ERR_USER_DIRECTORY_NOT_SET (p. 1377)	
PRL_ERR_USER_IS_ALREADY_LOGGED (p. 1377)	
PRL_ERR_USER_NO_AUTH_TO_CREATE_ROOT_VM_DIR (p. 1377)	
PRL_ERR_USER_NO_AUTH_TO_CREATE_VM_IN_DIR (p. 1378)	
PRL_ERR_USER_NO_AUTH_TO_EDIT_SERVER_SETTINGS (p. 1378)	
PRL_ERR_USER_NO_AUTH_TO_EDIT_VM (p. 1378)	
PRL_ERR_USER_NO_AUTH_TO_SAVE_BACKUP_FILES (p. 1378)	
PRL_ERR_USER_NO_AUTH_TO_SAVE_FILES (p. 1379)	
PRL_ERR_USER_NOT_FOUND (p. 1379)	
PRL_ERR_USER_OPERATION_NOT_AUTHORISED (p. 1379)	
PRL_ERR_USER_PROFILE_WAS_CHANGED (p. 1379)	
PRL_ERR_VA_CONFIG (p. 1379)	• Virtual Application Configuration errors +++ ALIGNED VALUE TO 18000
PRL_ERR_VCMMD_NO_CONNECTION (p. 1380)	
PRL_ERR_VM_ABORT (p. 1380)	
PRL_ERR_VM_ALLOC_MEM_DRV_NOT_STARTED (p. 1380)	
PRL_ERR_VM_ALLOC_VM_MEMORY (p. 1380)	
PRL_ERR_VM_ALREADY_CONNECTED (p. 1381)	
PRL_ERR_VM_ALREADY_CREATED (p. 1381)	
PRL_ERR_VM_ALREADY_REGISTERED (p. 1381)	
PRL_ERR_VM_ALREADY_REGISTERED_UNIQUE_PARAMS (p. 1381)	
PRL_ERR_VM_ALREADY_REGISTERED_VM_NAME (p. 1381)	
PRL_ERR_VM_ALREADY_REGISTERED_VM_PATH (p. 1382)	

PRL_ERR_VM_ALREADY_REGISTERED_VM_UUID (p. 1382)	
PRL_ERR_VM_ALREADY_RESUMED (p. 1382)	
PRL_ERR_VM_ALREADY_RUNNING (p. 1382)	
PRL_ERR_VM_ALREADY_SUSPENDED (p. 1383)	
PRL_ERR_VM_ANOTHER_TOOLS_IN_USE (p. 1383)	
PRL_ERR_VM_APPLY_CHANGES_PENDING (p. 1383)	
PRL_ERR_VM_APPLY_CONFIG_FAILED (p. 1383)	
PRL_ERR_VM_APPLY_CONFIG_NEEDS_REBOOT (p. 1383)	
PRL_ERR_VM_ASYNC_CD_CONSTRUCTOR (p. 1384)	
PRL_ERR_VM_AUTO_COMPRESS_TASK_ALREADY_RUN (p. 1384)	
PRL_ERR_VM_BACKUP_HDD_IMAGE_OUT_OF_BUNDLE (p. 1384)	
PRL_ERR_VM_BACKUP_INVALID_DISK_TYPE (p. 1384)	
PRL_ERR_VM_BAD_OS_TYPE (p. 1385)	
PRL_ERR_VM_CANT_CLONE_HDD_MISSING (p. 1385)	
PRL_ERR_VM_CANT_CLONE_RUNNING (p. 1385)	
PRL_ERR_VM_CANT_DELETE_RUNNING (p. 1385)	
PRL_ERR_VM_CANT_UNREG_RUNNING (p. 1385)	
PRL_ERR_VM_COMPACT_HARD_DISK_FAILED (p. 1386)	
PRL_ERR_VM_COMPACT_NOT_SUPPORTED_GUEST_OS (p. 1386)	
PRL_ERR_VM_COMPACT_PROCESSING (p. 1386)	• Optimization of virtual HDD Devices error +++ ALIGNED VALUE TO 28000
PRL_ERR_VM_CONF_CHANGED_INVALID_ARG (p. 1386)	
PRL_ERR_VM_CONF_CHANGED_NOT_CREATED (p. 1387)	
PRL_ERR_VM_CONF_CHANGED_NOT_STARTED (p. 1387)	
PRL_ERR_VM_CONF_CHANGED_UNSUPPORTED_DEV (p. 1387)	
PRL_ERR_VM_CONFIG_ALREADY_EXISTS (p. 1387)	
PRL_ERR_VM_CONFIG_CAN_BE_RESTORED (p. 1387)	
PRL_ERR_VM_CONFIG_DOESNT_EXIST (p. 1388)	
PRL_ERR_VM_CONFIG_INVALID_SERVER_UUID (p. 1388)	
PRL_ERR_VM_CONFIG_INVALID_VM_UUID (p. 1388)	
PRL_ERR_VM_CONFIG_IS_ALREADY_VALID (p. 1388)	
PRL_ERR_VM_CONFIG_WAS_CHANGED (p. 1389)	
PRL_ERR_VM_COULDNT_BE_STARTED_UNDER_SPECIFIED_USER (p. 1389)	
PRL_ERR_VM_CREATE_HDD_IMG_INVALID_ARG (p. 1389)	
PRL_ERR_VM_CREATE_HDD_IMG_INVALID_CREATE (p. 1389)	
PRL_ERR_VM_CREATE_INVALID_ARG (p. 1390)	
PRL_ERR_VM_CREATE_SNAPSHOT_FAILED (p. 1390)	
PRL_ERR_VM_DELETE_STATE_FAILED (p. 1390)	
PRL_ERR_VM_DEV_CHANGE_MEDIA_FAILED (p. 1390)	
PRL_ERR_VM_DEV_CONNECT_FAILED (p. 1390)	
PRL_ERR_VM_DEV_DISCONNECT_FAILED (p. 1391)	
PRL_ERR_VM_DEVICES_INITIALIZATION_FAILED (p. 1391)	
PRL_ERR_VM_DEVICES_TERMINATION_FAILED (p. 1391)	

PRL_ERR_VM_DIR_CONFIG_ALREADY_EXISTS (p. 1391)	
PRL_ERR_VM_DIR_FILE_NOT_SET (p. 1391)	
PRL_ERR_VM_DIRECTORY_FOLDER_DOESNT_EXIST (p. 1392)	
PRL_ERR_VM_DIRECTORY_NOT_EXIST (p. 1392)	
PRL_ERR_VM_DIRECTORY_NOT_INITIALIZED (p. 1392)	
PRL_ERR_VM_DOES_NOT_STOPPED (p. 1392)	
PRL_ERR_VM_DVD_DISCONNECT_FAILED_LOCKED (p. 1392)	
PRL_ERR_VM_EDIT_UNABLE_CHANGE_IP_ADDRESS (p. 1393)	
PRL_ERR_VM_EDIT_UNABLE_CONVERT_TO_TEMPLATE (p. 1393)	
PRL_ERR_VM_EDIT_UNABLE_SWITCH_OFF_UNDO_DISKS_MODE (p. 1393)	
PRL_ERR_VM_EDIT_UNABLE_SWITCH_ON_UNDO_DISKS_MODE (p. 1393)	
PRL_ERR_VM_EMPTY_NAME_OF_CLONE (p. 1394)	
PRL_ERR_VM_EXEC_GUEST_TOOL_NOT_AVAILABLE (p. 1394)	
PRL_ERR_VM_EXEC_PROGRAM_NOT_FOUND (p. 1394)	
PRL_ERR_VM_EXPIRATION_HOST_DATETIME_SKEWED (p. 1394)	
PRL_ERR_VM_EXPIRATION_OFFLINE_GRACE_PERIOD_EXPIRED (p. 1395)	
PRL_ERR_VM_EXPIRATION_TIME_CHECK_INTERVAL_OUT_OF_RANGE (p. 1395)	
PRL_ERR_VM_EXPIRATION_VM_IS_ABOUT_TO_EXPIRE (p. 1395)	
PRL_ERR_VM_EXPIRATION_VM_IS_IN_OFFLINE_GRACE_PERIOD (p. 1395)	
PRL_ERR_VM_FILES_ALREADY_REMOVED (p. 1396)	
PRL_ERR_VM_FREE_VM_MEMORY (p. 1396)	
PRL_ERR_VM_GET_CONFIG_FAILED (p. 1396)	
PRL_ERR_VM_GET_CONFIG_INVALID_ARG (p. 1396)	
PRL_ERR_VM_GET_CONFIG_NOT_CONFIGURED (p. 1397)	
PRL_ERR_VM_GET_DEVICE_STATE_FAILED (p. 1397)	
PRL_ERR_VM_GET_HDD_IMG_INVALID_ARG (p. 1397)	
PRL_ERR_VM_GET_HDD_IMG_NOT_OPEN (p. 1397)	
PRL_ERR_VM_GET_INDICATORS_INVALID_ARG (p. 1397)	
PRL_ERR_VM_GET_INDICATORS_NOT_CREATED (p. 1398)	
PRL_ERR_VM_GET_INDICATORS_NOT_RUNNING (p. 1398)	
PRL_ERR_VM_GET_PROBLEM_REPORT_FAILED (p. 1398)	
PRL_ERR_VM_GET_SCR_SIZE_NOT_CREATED (p. 1398)	
PRL_ERR_VM_GET_SCR_UPDATED_NOT_CREATED (p. 1398)	
PRL_ERR_VM_GET_SCR_UPDATED_NOT_RUNNING (p. 1399)	
PRL_ERR_VM_GET_STATUS_FAILED (p. 1399)	
PRL_ERR_VM_GET_STATUS_INVALID_ARG (p. 1399)	
PRL_ERR_VM_GUEST_SESSION_EXPIRED (p. 1399)	
PRL_ERR_VM_GUESTMEM_FAIL (p. 1399)	
PRL_ERR_VM_HARD_DISK_NOT_COMPACTABLE (p. 1400)	
PRL_ERR_VM_HDD_DISCONNECT_FAILED_LOCKED (p. 1400)	
PRL_ERR_VM_HDD_SIZE (p. 1400)	
PRL_ERR_VM_HOME_PATH_IS_NOT_EMPTY (p. 1400)	
PRL_ERR_VM_HYPERVISOR_HVT_ENABLED_ON_HYPERSWITCH (p. 1401)	
PRL_ERR_VM_HYPERVIZOR_COMM (p. 1401)	
PRL_ERR_VM_IN_FROZEN_STATE (p. 1401)	

PRL_ERR_VM_IN_HDD_CONSTRUCTOR (p. 1401)	
PRL_ERR_VM_IN_WAKING_UP_STATE (p. 1402)	
PRL_ERR_VM_INIT_MONITOR (p. 1402)	
PRL_ERR_VM_INIT_VCPU (p. 1402)	
PRL_ERR_VM_INTERACT_PRLS_DRIVER (p. 1402)	
PRL_ERR_VM_INTERNAL_OS_ERROR (p. 1402)	
PRL_ERR_VM_INVALID_SWAP_REGION (p. 1403)	
PRL_ERR_VM_IS_EXCLUSIVELY_LOCKED (p. 1403)	
PRL_ERR_VM_IS_NOT_LOCKED (p. 1403)	
PRL_ERR_VM_IS_NOT_SUSPENDED (p. 1403)	
PRL_ERR_VM_LIFETIME_IS_EXPIRED (p. 1403)	
PRL_ERR_VM_LOAD_BINARY_FAILED (p. 1404)	
PRL_ERR_VM_LOAD_MONITOR (p. 1404)	
PRL_ERR_VM_LOCKED_CTL_FOR_BACKUP (p. 1404)	
PRL_ERR_VM_LOCKED_FOR_BACKUP (p. 1404)	
PRL_ERR_VM_LOCKED_FOR_CHANGE_FIREWALL (p. 1404)	
PRL_ERR_VM_LOCKED_FOR_CHANGE_PASSWORD (p. 1405)	
PRL_ERR_VM_LOCKED_FOR_CLONE (p. 1405)	Dispatcher 'locked' error codes +++ ALIGNED VALUE TO 38100
PRL_ERR_VM_LOCKED_FOR_COPY_IMAGE (p. 1405)	
PRL_ERR_VM_LOCKED_FOR_CREATE_SNAPSHOT (p. 1405)	
PRL_ERR_VM_LOCKED_FOR_DECRYPT (p. 1405)	
PRL_ERR_VM_LOCKED_FOR_DELETE (p. 1406)	
PRL_ERR_VM_LOCKED_FOR_DELETE_TO_SNAPSHOT (p. 1406)	
PRL_ERR_VM_LOCKED_FOR_DISK_COMPACT (p. 1406)	
PRL_ERR_VM_LOCKED_FOR_DISK_CONVERT (p. 1406)	
PRL_ERR_VM_LOCKED_FOR_DISK_RESIZE (p. 1407)	
PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT (p. 1407)	
PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT_WITH_RENAME (p. 1407)	
PRL_ERR_VM_LOCKED_FOR_ENCRYPT (p. 1407)	
PRL_ERR_VM_LOCKED_FOR_EXECUTE (p. 1408)	
PRL_ERR_VM_LOCKED_FOR_EXECUTE_EX (p. 1408)	
PRL_ERR_VM_LOCKED_FOR_INTERNAL_REASON (p. 1408)	
PRL_ERR_VM_LOCKED_FOR_MIGRATE (p. 1408)	
PRL_ERR_VM_LOCKED_FOR_MOVE (p. 1408)	
PRL_ERR_VM_LOCKED_FOR_REMOVE_PROTECTION (p. 1409)	
PRL_ERR_VM_LOCKED_FOR_RESTORE_FROM_BACKUP (p. 1409)	
PRL_ERR_VM_LOCKED_FOR_SET_PROTECTION (p. 1409)	
PRL_ERR_VM_LOCKED_FOR_SWITCH_TO_SNAPSHOT (p. 1409)	
PRL_ERR_VM_LOCKED_FOR_UNREGISTER (p. 1410)	
PRL_ERR_VM_LOCKED_FOR_UPDATE_SECURITY (p. 1410)	
PRL_ERR_VM_MAP_VM_MEM0 (p. 1410)	
PRL_ERR_VM_MEMORY_SWAPPING_IN_PROGRESS (p. 1410)	
PRL_ERR_VM_MIGRATE_ACCESS_TO_VM_DENIED (p. 1410)	
PRL_ERR_VM_MIGRATE_BREAK_BY_DISK_CONDITION (p. 1411)	

PRL_ERR_VM_MIGRATE_CANNOT_CREATE_DIRECTORY (p. 1411)	
PRL_ERR_VM_MIGRATE_CANNOT_REMOTE_CLONE_SHARED_VM (p. 1411)	
PRL_ERR_VM_MIGRATE_CHECKING_PRECONDITIONS_FAILED (p. 1411)	VM migration errors codes set +++ ALIGNED VALUE TO 31000
PRL_ERR_VM_MIGRATE_CONTINUE_START_FAILED (p. 1412)	
PRL_ERR_VM_MIGRATE_COULDNT_DETACH_TARGET_CONNECTION (p. 1412)	
PRL_ERR_VM_MIGRATE_DEVICE_IMAGE_OUT_OF_BUNDLE (p. 1412)	
PRL_ERR_VM_MIGRATE_ERROR_DELETE_VM (p. 1412)	
PRL_ERR_VM_MIGRATE_ERROR_UNREGISTER_VM (p. 1413)	
PRL_ERR_VM_MIGRATE_EXT_DISK_DIR_ALREADY_EXISTS_ON_TARGET (p. 1413)	
PRL_ERR_VM_MIGRATE_EXTERNAL_DISKS_NOT_SUPPORTED (p. 1413)	
PRL_ERR_VM_MIGRATE_FLOPPY_DISK_IS_ABSENT_ON_TARGET (p. 1413)	
PRL_ERR_VM_MIGRATE_INVALID_DISK_TYPE (p. 1414)	
PRL_ERR_VM_MIGRATE_NETWORK_ADAPTER_IS_ABSENT_ON_TARGET (p. 1414)	
PRL_ERR_VM_MIGRATE_NETWORK_SHARE_IS_ABSENT_ON_TARGET (p. 1414)	
PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET (p. 1415)	
PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET_SHORT (p. 1415)	
PRL_ERR_VM_MIGRATE_NOT_ENOUGH_CPUS_ON_TARGET (p. 1415)	
PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_SOURCE (p. 1416)	
PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_TARGET (p. 1416)	
PRL_ERR_VM_MIGRATE_OPTICAL_DISK_IS_ABSENT_ON_TARGET (p. 1416)	
PRL_ERR_VM_MIGRATE_OUT_OF_MEMORY_ON_TARGET (p. 1416)	
PRL_ERR_VM_MIGRATE_PARALLEL_PORT_IS_ABSENT_ON_TARGET (p. 1417)	
PRL_ERR_VM_MIGRATE_REGISTER_VM_FAILED (p. 1417)	
PRL_ERR_VM_MIGRATE_REMOTE_DEVICE_IS_ATTACHED (p. 1417)	
PRL_ERR_VM_MIGRATE_RESUME_FAILED (p. 1417)	
PRL_ERR_VM_MIGRATE_RESUME_VM_FAILED (p. 1418)	
PRL_ERR_VM_MIGRATE_SERIAL_PORT_IS_ABSENT_ON_TARGET (p. 1418)	
PRL_ERR_VM_MIGRATE_SOUND_DEVICE_IS_ABSENT_ON_TARGET (p. 1418)	
PRL_ERR_VM_MIGRATE_STORAGE_INFO_PARSE (p. 1418)	
PRL_ERR_VM_MIGRATE_SUSPEND_VM_FAILED (p. 1419)	
PRL_ERR_VM_MIGRATE_TARGET_INSIDE_SHARED_VM_PRIVATE (p. 1419)	
PRL_ERR_VM_MIGRATE_TARGET_VM_HOME_PATH_NOT_EXISTS (p. 1419)	
PRL_ERR_VM_MIGRATE_TO_THE_SAME_NODE (p. 1419)	
PRL_ERR_VM_MIGRATE_UNSUITABLE_VM_STATE (p. 1420)	
PRL_ERR_VM_MIGRATE_USB_CONTROLLER_IS_ABSENT_ON_TARGET (p. 1420)	
PRL_ERR_VM_MIGRATE_VM_ALREADY_EXISTS_ON_TARGET (p. 1420)	
PRL_ERR_VM_MIGRATE_VM_ALREADY_MIGRATE_ON_TARGET (p. 1420)	
PRL_ERR_VM_MIGRATE_VM_HOME_ALREADY_EXISTS_ON_TARGET (p. 1421)	
PRL_ERR_VM_MIGRATE_VM_UUID_ALREADY_EXISTS_ON_TARGET (p. 1421)	
PRL_ERR_VM_MIGRATE_WARM_MODE_NOT_SUPPORTED (p. 1421)	
PRL_ERR_VM_MONITOR (p. 1421)	
PRL_ERR_VM_MOUNT (p. 1422)	VM hdd mount/umount +++ ALIGNED VALUE TO 46600
PRL_ERR_VM_MUST_BE_STOPPED_BEFORE_RENAMING (p. 1422)	
PRL_ERR_VM_MUST_BE_STOPPED_FOR_CHANGE_DEVICES (p. 1422)	

PRL_ERR_VM_NAME_IS_EMPTY (p. 1422)	
PRL_ERR_VM_NOT_CREATED (p. 1423)	
PRL_ERR_VM_OPERATION_FAILED (p. 1423)	
PRL_ERR_VM_PAUSE_ALREADY_PAUSED (p. 1423)	
PRL_ERR_VM_PAUSE_FAILED (p. 1423)	
PRL_ERR_VM_PAUSE_NOT_CREATED (p. 1423)	
PRL_ERR_VM_PAUSE_NOT_STARTED (p. 1424)	
PRL_ERR_VM_POWER_OFF_FAILED (p. 1424)	
PRL_ERR_VM_POWER_ON_FAILED (p. 1424)	
PRL_ERR_VM_PRLS_DESCTOP_ALLOC_PHYS_MEM (p. 1424)	
PRL_ERR_VM_PROCESS_IS_NOT_STARTED (p. 1424)	
PRL_ERR_VM_PROCESS_IS_NOT_STARTED_BY_WRONG_HOST_SW (p. 1425)	
PRL_ERR_VM_REQUEST_NOT_SUPPORTED (p. 1425)	
PRL_ERR_VM_RESET_FAILED (p. 1425)	
PRL_ERR_VM_RESTART_GUEST_FAILED (p. 1425)	
PRL_ERR_VM_RESTART_NOT_CREATED (p. 1426)	
PRL_ERR_VM_RESTART_NOT_STARTED (p. 1426)	
PRL_ERR_VM_RESTORE_STATE_FAILED (p. 1426)	
PRL_ERR_VM_RESTORE_STATE_FAILED_VM_STOPPED (p. 1426)	
PRL_ERR_VM_RESUME_FAILED (p. 1426)	
PRL_ERR_VM_RESUME_INV_SAV_VERSION_CANCEL_RESUME (p. 1427)	
PRL_ERR_VM_SAVE_STATE_FAILED (p. 1427)	
PRL_ERR_VM_SEND_ABS_MOUSE_NOT_CREATED (p. 1427)	
PRL_ERR_VM_SEND_ABS_MOUSE_NOT_RUNNING (p. 1427)	
PRL_ERR_VM_SEND_ABS_MOUSE_PAUSED (p. 1428)	
PRL_ERR_VM_SEND_KEYBOARD_NOT_CREATED (p. 1428)	
PRL_ERR_VM_SEND_KEYBOARD_NOT_RUNNING (p. 1428)	
PRL_ERR_VM_SEND_KEYBOARD_PAUSED (p. 1428)	
PRL_ERR_VM_SEND_REL_MOUSE_NOT_CREATED (p. 1428)	
PRL_ERR_VM_SEND_REL_MOUSE_NOT_RUNNING (p. 1429)	
PRL_ERR_VM_SEND_REL_MOUSE_PAUSED (p. 1429)	
PRL_ERR_VM_SET_CONFIG_FAILED (p. 1429)	
PRL_ERR_VM_SET_CONFIG_INVALID_ARG (p. 1429)	
PRL_ERR_VM_SET_LICENSE_FAILED (p. 1429)	
PRL_ERR_VM_SET_VISIBLE_NOT_CREATED (p. 1430)	
PRL_ERR_VM_SHUTDOWN_FAILED (p. 1430)	
PRL_ERR_VM_SHUTDOWN_HIBERNATE_FAILED (p. 1430)	
PRL_ERR_VM_SHUTDOWN_HIBERNATE_NOT_SUPPORTED (p. 1430)	
PRL_ERR_VM_SHUTDOWN_MACHINE_LOCKED (p. 1431)	
PRL_ERR_VM_SHUTDOWN_SUSPEND_NOT_SUPPORTED (p. 1431)	
PRL_ERR_VM_SNAPSHOT_CHANGED_VM_CONFIG (p. 1431)	
PRL_ERR_VM_SNAPSHOT_IN_SAFE_MODE (p. 1431)	
PRL_ERR_VM_SNAPSHOT_IN_UNDO_DISKS_MODE (p. 1431)	
PRL_ERR_VM_SNAPSHOT_NOT_FOUND (p. 1432)	
PRL_ERR_VM_SNAPSHOTS_CONFIG_NOT_FOUND (p. 1432)	

PRL_ERR_VM_SPECIFY_GUEST_INSTALL_FILES (p. 1432)	
PRL_ERR_VM_START_FAILED (p. 1432)	
PRL_ERR_VM_START_NOT_CONFIGURED (p. 1433)	
PRL_ERR_VM_START_NOT_CREATED (p. 1433)	
PRL_ERR_VM_STOP_NOT_CREATED (p. 1433)	
PRL_ERR_VM_SUSPEND_CHANGED_VM_CONFIG (p. 1433)	
PRL_ERR_VM_SUSPEND_FAILED (p. 1433)	
PRL_ERR_VM_TO_REGISTER_IS_NOT_SPECIFIED (p. 1434)	
PRL_ERR_VM_TOOL_NOT_AVAILABLE (p. 1434)	
PRL_ERR_VM_TOOLS_CANT_PARSE_UPDATE_PARAMETERS (p. 1434)	VM tools erros codes set +++ ALIGNED VALUE TO 33000
PRL_ERR_VM_TOOLS_CANT_UPDATE_WITHOUT_RESTART (p. 1434)	
PRL_ERR_VM_UNABLE_ALLOC_MEM (p. 1435)	
PRL_ERR_VM_UNABLE_ALLOC_MEM_MONITOR (p. 1435)	
PRL_ERR_VM_UNABLE_CREATE_TIMER (p. 1435)	
PRL_ERR_VM_UNABLE_GET_GUEST_CPU (p. 1435)	
PRL_ERR_VM_UNABLE_OPEN_DISK_IMAGE (p. 1435)	
PRL_ERR_VM_UNABLE_SEND_REQUEST (p. 1436)	
PRL_ERR_VM_UNABLE_TO_OPEN_VIRTUAL_BRIDGE (p. 1436)	
PRL_ERR_VM_UNDEFINED_API_CALL (p. 1436)	
PRL_ERR_VM_UNMOUNT (p. 1436)	
PRL_ERR_VM_UNPAUSE_FAILED (p. 1436)	
PRL_ERR_VM_UPDATE_SNAPSHOT_DATA_FAILED (p. 1437)	
PRL_ERR_VM_USER_AUTHENTICATION_FAILED (p. 1437)	
PRL_ERR_VM_UUID_EMPTY (p. 1437)	
PRL_ERR_VM_UUID_NOT_FOUND (p. 1437)	
PRL_ERR_VM_VALLOC_MON_BODY (p. 1438)	
PRL_ERR_VM_VALLOC_PE_IMG (p. 1438)	
PRL_ERR_VM_VIEW_SCR_INVALID_ARG (p. 1438)	
PRL_ERR_VM_VIEW_SCR_NOT_CREATED (p. 1438)	
PRL_ERR_VM_VIEW_SCR_NOT_RUNNING (p. 1438)	
PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN (p. 1439)	Unused PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN
PRL_ERR_VMCONF_BOOT_OPTION_DEVICE_NOT_EXISTS (p. 1439)	
PRL_ERR_VMCONF_BOOT_OPTION_DUPLICATE_DEVICE (p. 1439)	
PRL_ERR_VMCONF_BOOT_OPTION_INVALID_DEVICE_TYPE (p. 1439)	ALIGNED VALUE TO 27050
PRL_ERR_VMCONF_BOOTCAMP_HARD_DISK_SMART_GUARD_NOT_ALLOW (p. 1440)	
PRL_ERR_VMCONF_BOOTCAMP_HARD_SNAPSHOTS_NOT_ALLOW (p. 1440)	
PRL_ERR_VMCONF_BOOTCAMP_HARD_UNDO_DISKS_NOT_ALLOW (p. 1440)	
PRL_ERR_VMCONF_BOOTCAMP_SAFE_MODE_NOT_ALLOW (p. 1441)	
PRL_ERR_VMCONF_CD_DVD_ROM_DUPLICATE_SYS_NAME (p. 1441)	
PRL_ERR_VMCONF_CD_DVD_ROM_IMAGE_IS_NOT_EXIST (p. 1441)	
PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_OLD_CHIPSET (p. 1441)	

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_UNSUPPORTED_OS (p. 1442)	
PRL_ERR_VMCONF_CD_DVD_ROM_SYS_NAME_IS_EMPTY (p. 1442)	ALIGNED VALUE TO 27400
PRL_ERR_VMCONF_CD_DVD_ROM_URL_FORMAT_SYS_NAME (p. 1442)	
PRL_ERR_VMCONF_CPU_COUNT_MORE_HOST_CPU_COUNT (p. 1443)	
PRL_ERR_VMCONF_CPU_COUNT_MORE_MAX_CPU_COUNT (p. 1443)	
PRL_ERR_VMCONF_CPU_MASK_INVALID (p. 1443)	
PRL_ERR_VMCONF_CPU_MASK_INVALID_CPU_NUM (p. 1443)	
PRL_ERR_VMCONF_CPU_NODE_MASK_INVALID (p. 1444)	
PRL_ERR_VMCONF_CPU_ZERO_COUNT (p. 1444)	ALIGNED VALUE TO 27200
PRL_ERR_VMCONF_CPULIMIT_NOT_SUPPORTED (p. 1444)	
PRL_ERR_VMCONF_CPUUNITS_NOT_SUPPORTED (p. 1444)	
PRL_ERR_VMCONF_DESKTOP_MODE_REMOTE_DEVICES (p. 1445)	
PRL_ERR_VMCONF_DUPLICATE_IP_ADDRESS (p. 1445)	
PRL_ERR_VMCONF_EFI_UNSUPPORTED_GUEST (p. 1445)	
PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_EXIST (p. 1445)	
PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_VALID (p. 1446)	
PRL_ERR_VMCONF_FLOPPY_DISK_IS_NOT_ACCESSIBLE (p. 1446)	
PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_HAS_INVALID_SYMBOL (p. 1446)	
PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_IS_EMPTY (p. 1446)	ALIGNED VALUE TO 27350
PRL_ERR_VMCONF_FLOPPY_DISK_URL_FORMAT_SYS_NAME (p. 1447)	
PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_CANNOT_BE_ADDED (p. 1447)	ALIGNED VALUE TO 27760
PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_DUPLICATE_IN_ANOTHER_VM (p. 1447)	
PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_CONNECTED (p. 1448)	
PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_FOUND (p. 1448)	
PRL_ERR_VMCONF_GENERIC_PCI_DUPLICATE_SYS_NAME (p. 1448)	
PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE (p. 1448)	PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE is unused
PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_NOT_SINGLE (p. 1449)	
PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_WRONG_COUNT (p. 1449)	
PRL_ERR_VMCONF_GENERIC_PCI_WRONG_DEVICE (p. 1449)	
PRL_ERR_VMCONF_HARD_DISK_DUPLICATE_SYS_NAME (p. 1450)	
PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_EXIST (p. 1450)	
PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_VALID (p. 1450)	
PRL_ERR_VMCONF_HARD_DISK_MISS_BOOTCAMP_PARTITION (p. 1450)	
PRL_ERR_VMCONF_HARD_DISK_NOT_ENOUGH_SPACE_FOR_ENCRYPT_DISK (p. 1451)	
PRL_ERR_VMCONF_HARD_DISK_SERIAL_IS_NOT_VALID (p. 1451)	
PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_OLD_CHIPSET (p. 1451)	
PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_UNSUPPORTED_OS (p. 1451)	
PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_HAS_INVALID_SYMBOL (p. 1452)	
PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_IS_EMPTY (p. 1452)	ALIGNED VALUE TO 27450
PRL_ERR_VMCONF_HARD_DISK_UNABLE_DELETE_DISK_WITH_SNAPSHOTS (p. 1452)	
PRL_ERR_VMCONF_HARD_DISK_URL_FORMAT_SYS_NAME (p. 1453)	
PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_BOOTCAMP_PARTITION (p. 1453)	

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_FOR_ENCRYPTRED_VM (p. 1453)	
PRL_ERR_VMCONF_IDE_DEVICES_COUNT_OUT_OF_RANGE (p. 1453)	ALIGNED VALUE TO 27700
PRL_ERR_VMCONF_IDE_DEVICES_DUPLICATE_STACK_INDEX (p. 1454)	
PRL_ERR_VMCONF_INCOMPAT_HARD_DISK_SMART_GUARD_NOT_ALLOW (p. 1454)	
PRL_ERR_VMCONF_INCOMPAT_HARD_UNDO_DISKS_NOT_ALLOW (p. 1454)	
PRL_ERR_VMCONF_INCOMPAT_SAFE_MODE_NOT_ALLOW (p. 1455)	
PRL_ERR_VMCONF_INVALID_DEVICE_MAIN_INDEX (p. 1455)	
PRL_ERR_VMCONF_IOLIMIT_NOT_SUPPORTED (p. 1455)	
PRL_ERR_VMCONF_IOPRIO_NOT_SUPPORTED (p. 1455)	
PRL_ERR_VMCONF_IOPSLIMIT_NOT_SUPPORTED (p. 1456)	
PRL_ERR_VMCONF_MAIN_MEMORY_MAX_BALLOON_SIZE_MORE_100_PERCENT (p. 1456)	
PRL_ERR_VMCONF_MAIN_MEMORY_MQ_INVALID_RANGE (p. 1456)	
PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_LESS_VMM_OVERHEAD_VALUE (p. 1456)	
PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_OUT_OF_RANGE (p. 1457)	
PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_OUT_OF_RANGE (p. 1457)	
PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_ZERO (p. 1457)	
PRL_ERR_VMCONF_MAIN_MEMORY_NOT_4_RATIO_SIZE (p. 1457)	
PRL_ERR_VMCONF_MAIN_MEMORY_OUT_OF_RANGE (p. 1458)	
PRL_ERR_VMCONF_MAIN_MEMORY_SIZE_NOT_EAQUAL_RECOMMENDED (p. 1458)	
PRL_ERR_VMCONF_MAIN_MEMORY_ZERO_SIZE (p. 1458)	ALIGNED VALUE TO 27250
PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTART (p. 1458)	
PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTOP (p. 1459)	
PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_KEEP_VM_ALIVE_ON_GUI_EXIT (p. 1459)	
PRL_ERR_VMCONF_NEED_MORE_MEMORY_TO_ENABLE_HOTPLUG (p. 1459)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_BROADCAST_IP_ADDRESS (p. 1460)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_IP_ADDRESS (p. 1460)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_MAC_ADDRESS (p. 1460)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_ETHLIST_CREATE_ERROR (p. 1461)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_GATEWAY_NOT_IN_SUBNET (p. 1461)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_GUEST_TOOLS_NOT_AVAILABLE (p. 1461)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_BOUND_INDEX (p. 1462)	ALIGNED VALUE TO 27500
PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_DNS_IP_ADDRESS (p. 1462)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_GATEWAY_IP_ADDRESS (p. 1462)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_IP_ADDRESS (p. 1463)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_MAC_ADDRESS (p. 1463)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_SEARCH_DOMAIN_NAME (p. 1463)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_MULTICAST_IP_ADDRESS (p. 1464)	
PRL_ERR_VMCONF_NETWORK_ADAPTER_ROUTED_NO_STATIC_ADDRESS (p. 1464)	
PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_SMART_GUARD (p. 1464)	
PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_UNDO_DISKS (p. 1465)	

PRL_ERR_VMCONF_NO_CONFIGURED_TOTALRATE_FOR_NETWORK_CLASS (p. 1465)	
PRL_ERR_VMCONF_NO_HD_IMAGES_IN_SAFE_MODE (p. 1465)	
PRL_ERR_VMCONF_NO_HD_IMAGES_IN_UNDO_DISKS_MODE (p. 1465)	
PRL_ERR_VMCONF_NO_IP_ADDRESSES_SPECIFIED_FOR_OFFLINE_MANAGEMENT (p. 1466)	ALIGNED VALUE TO 27770
PRL_ERR_VMCONF_NO_SMART_GUARD_WITH_UNDO_DISKS (p. 1466)	
PRL_ERR_VMCONF_PARALLEL_PORT_IMAGE_IS_NOT_EXIST (p. 1466)	
PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_HAS_INVALID_SYMBOL (p. 1467)	
PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_IS_EMPTY (p. 1467)	ALIGNED VALUE TO 27650
PRL_ERR_VMCONF_PARALLEL_PORT_URL_FORMAT_SYS_NAME (p. 1467)	
PRL_ERR_VMCONF_REAL_HARD_SAFE_MODE_NOT_ALLOW (p. 1468)	
PRL_ERR_VMCONF_REAL_HARD_UNDO_DISKS_NOT_ALLOW (p. 1468)	
PRL_ERR_VMCONF_REMOTE_DISPLAY_EMPTY_PASSWORD (p. 1468)	
PRL_ERR_VMCONF_REMOTE_DISPLAY_HOST_IP_ADDRESS_IS_ZERO (p. 1469)	
PRL_ERR_VMCONF_REMOTE_DISPLAY_INVALID_HOST_IP_ADDRESS (p. 1469)	
PRL_ERR_VMCONF_REMOTE_DISPLAY_PASSWORD_TOO_LONG (p. 1469)	
PRL_ERR_VMCONF_REMOTE_DISPLAY_PORT_NUMBER_IS_ZERO (p. 1469)	ALIGNED VALUE TO 27100
PRL_ERR_VMCONF_RESTRICTED_CPU_COUNT (p. 1470)	
PRL_ERR_VMCONF_RESTRICTED_MEMORY_SIZE (p. 1470)	
PRL_ERR_VMCONF_RESTRICTED_OS_VERSION (p. 1470)	• Validation VM configuration error +++ ALIGNED VALUE TO 26500
PRL_ERR_VMCONF_RESTRICTED_SMART_GUARD (p. 1470)	
PRL_ERR_VMCONF_RESTRICTED_UNDO_DISKS (p. 1471)	
PRL_ERR_VMCONF_SATA_DEVICES_COUNT_OUT_OF_RANGE (p. 1471)	
PRL_ERR_VMCONF_SATA_DEVICES_DUPLICATE_STACK_INDEX (p. 1471)	
PRL_ERR_VMCONF_SCSI_BUSLOGIC_WITH_EFI_NOT_SUPPORTED (p. 1471)	
PRL_ERR_VMCONF_SCSI_DEVICES_COUNT_OUT_OF_RANGE (p. 1472)	ALIGNED VALUE TO 27750
PRL_ERR_VMCONF_SCSI_DEVICES_DUPLICATE_STACK_INDEX (p. 1472)	
PRL_ERR_VMCONF_SERIAL_PORT_IMAGE_IS_NOT_EXIST (p. 1472)	
PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_HAS_INVALID_SYMBOL (p. 1473)	
PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_IS_EMPTY (p. 1473)	ALIGNED VALUE TO 27600
PRL_ERR_VMCONF_SERIAL_PORT_URL_FORMAT_SYS_NAME (p. 1473)	
PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_NAME (p. 1474)	
PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_PATH (p. 1474)	
PRL_ERR_VMCONF_SHARED_FOLDERS_EMPTY_FOLDER_NAME (p. 1474)	ALIGNED VALUE TO 27150
PRL_ERR_VMCONF_SHARED_FOLDERS_INVALID_FOLDER_PATH (p. 1475)	
PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY (p. 1475)	+++ ALIGNED VALUE TO 27550 +++ Unused PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY
PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY (p. 1475)	Unused PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY
PRL_ERR_VMCONF_UNKNOWN_OS_TYPE (p. 1475)	

PRL_ERR_VMCONF_UNKNOWN_OS_VERSION (p. 1476)	
PRL_ERR_VMCONF_VALIDATION_FAILED (p. 1476)	ALIGNED VALUE TO 27000
PRL_ERR_VMCONF_VIDEO_MEMORY_OUT_OF_RANGE (p. 1476)	ALIGNED VALUE TO 27300
PRL_ERR_VMCONF_VIDEO_NOT_ENABLED (p. 1476)	
PRL_ERR_VMCONF_VIRTIO_BLOCK_DEVICES_DUPLICATE_STACK_INDEX (p. 1477)	
PRL_ERR_VMCONF_VM_NAME_HAS_INVALID_SYMBOL (p. 1477)	
PRL_ERR_VMCONF_VM_NAME_IS_EMPTY (p. 1477)	
PRL_ERR_VMCONF_VM_NAME_IS_TOO_LONG (p. 1477)	
PRL_ERR_VMDIR_INVALID_PATH (p. 1478)	
PRL_ERR_VMDIR_PATH_IS_NOT_ABSOLUTE (p. 1478)	
PRL_ERR_VMHELPER_CREATEVM_FAILED (p. 1478)	STW7 VmHelper errors +++ ALIGNED VALUE TO 41400
PRL_ERR_VMHELPER_DIR_PATH_INVALID (p. 1478)	
PRL_ERR_VMHELPER_DISK_MOUNT_FAILED (p. 1478)	
PRL_ERR_VMHELPER_DISK_UNMOUNT_FAILED (p. 1479)	
PRL_ERR_VMHELPER_GETTING_AVAILABLE_DRIVE_FAILED (p. 1479)	
PRL_ERR_VNC_SERVER_ALREADY_STARTED (p. 1479)	VNC server operations errors +++ ALIGNED VALUE TO 35000
PRL_ERR_VNC_SERVER_AUTOSET_PORT_FAILED (p. 1479)	
PRL_ERR_VNC_SERVER_DISABLED (p. 1480)	
PRL_ERR_VNC_SERVER_NOT_STARTED (p. 1480)	
PRL_ERR_VOLUME_LICENSE_EXCEEDED_LIMIT (p. 1480)	
PRL_ERR_VTD_ALREADY_HOOKED_FAILED (p. 1480)	
PRL_ERR_VTD_DEVICE_DOES_NOT_EXIST (p. 1480)	
PRL_ERR_VTD_DEVICE_NOT_MAPPED (p. 1481)	• VT-d +++ ALIGNED VALUE TO 12000
PRL_ERR_VTD_DEVICE_TROUBLESHOOT (p. 1481)	
PRL_ERR_VTD_HOOK_AFTER_INSTALL_NEED_REBOOT (p. 1481)	
PRL_ERR_VTD_HOOK_AFTER_REVERT_NEED_REBOOT (p. 1481)	
PRL_ERR_VTD_HOOK_DEVICE_CURRENTLY_IN_USE (p. 1482)	
PRL_ERR_VTD_HOOK_FAILED (p. 1482)	
PRL_ERR_VTD_HOOK_INSTALLATION_FAILED (p. 1482)	
PRL_ERR_VTD_HOOK_INVALID_CONFIG (p. 1482)	
PRL_ERR_VTD_HOOK_NEED_REBOOT_FAILED (p. 1482)	
PRL_ERR_VTD_HOOK_REVERT_FAILED (p. 1483)	
PRL_ERR_VTD_HOOK_UPDATE_SCRIPT_EXECUTE (p. 1483)	
PRL_ERR_VTD_INITIALIZATION_FAILED (p. 1483)	
PRL_ERR_VTD_WAIT_ASR (p. 1483)	
PRL_ERR_VTX_ENABLED_ONLY_IN_SMX (p. 1483)	
PRL_ERR_VZ_API_NOT_INITIALIZED (p. 1484)	
PRL_ERR_VZ_OPERATION_FAILED (p. 1484)	
PRL_ERR_VZ_OSTEMPLATE_NOT_FOUND (p. 1484)	
PRL_ERR_VZCTL_OPERATION_FAILED (p. 1484)	
PRL_ERR_VZLICENSE_ACTIVATION_KEY (p. 1484)	

PRL_ERR_VZLICENSE_CANCEL (p. 1485)	
PRL_ERR_VZLICENSE_EACCES (p. 1485)	
PRL_ERR_VZLICENSE_ERR_KA (p. 1485)	
PRL_ERR_VZLICENSE_EXIST (p. 1485)	
PRL_ERR_VZLICENSE_FATAL (p. 1485)	
PRL_ERR_VZLICENSE_INVAL (p. 1486)	
PRL_ERR_VZLICENSE_INVALID_HWID (p. 1486)	
PRL_ERR_VZLICENSE_IO (p. 1486)	
PRL_ERR_VZLICENSE_KA_ACTIVATION_LIMIT_REACHED (p. 1486)	
PRL_ERR_VZLICENSE_KA_HWID_DOES_NOT_MATCH (p. 1487)	
PRL_ERR_VZLICENSE_KA_LICENSE_EXPIRED (p. 1487)	
PRL_ERR_VZLICENSE_KA_LICENSE_IS_NOT_PROLONGATED_YET (p. 1487)	
PRL_ERR_VZLICENSE_KA_LICENSE_IS_TERMINATED (p. 1487)	
PRL_ERR_VZLICENSE_KA_LICENSE_IS_UP_TO_DATE (p. 1488)	
PRL_ERR_VZLICENSE_LOCK (p. 1488)	
PRL_ERR_VZLICENSE_NETWORK (p. 1488)	
PRL_ERR_VZLICENSE_NODATA (p. 1488)	
PRL_ERR_VZLICENSE_NOENT (p. 1488)	
PRL_ERR_VZLICENSE_NOINIT (p. 1489)	
PRL_ERR_VZLICENSE_NOMEM (p. 1489)	
PRL_ERR_VZLICENSE_NOTSUP (p. 1489)	
PRL_ERR_VZLICENSE_PROXY (p. 1489)	
PRL_ERR_VZLICENSE_PROXY_AUTH (p. 1490)	
PRL_ERR_VZLICENSE_TASK_ALREADY_RUN (p. 1490)	
PRL_ERR_VZLICENSE_TIMEOUT (p. 1490)	
PRL_ERR_VZLICENSE_UNSUPPORTED_APP_MODE (p. 1490)	
PRL_ERR_VZLICENSE_UNSUPPORTED_MODE (p. 1490)	
PRL_ERR_VZLICENSE_UNSUPPORTED_UPDATE_OP (p. 1491)	
PRL_ERR_VZLICENSE_VMS_LIMIT_EXCEEDED (p. 1491)	
PRL_ERR_VZLICENSE_WRONG (p. 1491)	
PRL_ERR_WEB_PORTAL_ACCEPTED (p. 1491)	
PRL_ERR_WEB_PORTAL_ACCOUNT_ALREADY_EXISTS (p. 1491)	
PRL_ERR_WEB_PORTAL_BAD_REQUEST (p. 1492)	errors
PRL_ERR_WEB_PORTAL_CONFLICT (p. 1492)	
PRL_ERR_WEB_PORTAL_CREATED (p. 1492)	
PRL_ERR_WEB_PORTAL_FORBIDDEN (p. 1492)	
PRL_ERR_WEB_PORTAL_INVALID_EMAIL (p. 1493)	
PRL_ERR_WEB_PORTAL_INVALID_PARAMETERS (p. 1493)	
PRL_ERR_WEB_PORTAL_INVALID_PASSWORD (p. 1493)	
PRL_ERR_WEB_PORTAL_NO_CONTENT (p. 1493)	
PRL_ERR_WEB_PORTAL_NOT_FOUND (p. 1493)	
PRL_ERR_WEB_PORTAL_OK (p. 1494)	success
PRL_ERR_WEB_PORTAL_SERVICE_UNAVAILABLE (p. 1494)	
PRL_ERR_WEB_PORTAL_SOCIAL_APP_DENIED (p. 1494)	
PRL_ERR_WEB_PORTAL_SOCIAL_CONFIRM_EMAIL_SENT (p. 1494)	

PRL_ERR_WEB_PORTAL_SOCIAL_PROHIBITED_FOR_BA (p. 1495)	
PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_BAD_RESPONSE (p. 1495)	
PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_UNAVAILABLE (p. 1495)	
PRL_ERR_WEB_PORTAL_SOCIAL_UNCONFIRMED_ACCOUNT (p. 1495)	
PRL_ERR_WEB_PORTAL_UNAUTHORIZED (p. 1496)	
PRL_ERR_WIN_AERO_DISABLED (p. 1496)	isn't used now
PRL_ERR_WINDOWS_EXPRESS_INSTALL_USER_NAME_EMPTY (p. 1496)	
PRL_ERR_WRITE_FAILED (p. 1496)	
PRL_ERR_WRONG_CONNECTION_SECURITY_LEVEL (p. 1497)	
PRL_ERR_WRONG_PROTOCOL_VERSION (p. 1497)	
PRL_ERR_WRONG_VM_STATE (p. 1497)	
PRL_ERR_WS_DISP_CONNECTION_CLOSED (p. 1497)	
PRL_ERR_X64GUEST_ON_X32HOST (p. 1497)	
PRL_ERR_X64GUEST_ON_X64VM_HVT_DISABLED (p. 1498)	
PRL_ERR_XML_WRITE_FILE (p. 1498)	
PRL_ERR_XMLRPC_INTERNAL_ERROR (p. 1498)	
PRL_ERR_XMLRPC_INVALID_CREDENTIALS (p. 1498)	
PRL_ERR_XMLRPC_INVALID_REQUEST_INFO (p. 1498)	
PRL_ERR_XMLRPC_LIMITS_EXHAUSTED (p. 1499)	
PRL_ERR_XMLRPC_WRONG_METHOD_CALL (p. 1499)	

PRL_ERR_ACCES_DENIED_FILE_TO_PARENT_PARENT_DIR Macro

Syntax

```
#define PRL_ERR_ACCES_DENIED_FILE_TO_PARENT_PARENT_DIR PRL_RESULT_DECLARE_ERROR(100)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_DENIED Macro

Syntax

```
#define PRL_ERR_ACCESS_DENIED PRL_RESULT_DECLARE_ERROR(5)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_DENIED_TO_CHANGE_PERMISSIONS Macro

Syntax

```
#define PRL_ERR_ACCESS_DENIED_TO_CHANGE_PERMISSIONS PRL_RESULT_DECLARE_ERROR(381)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_DENIED_TO_DISK_IMAGE Macro

Syntax

```
#define PRL_ERR_ACCESS_DENIED_TO_DISK_IMAGE PRL_RESULT_DECLARE_ERROR(78)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_DENINED_TO_RUN_WIZARD Macro

Syntax

```
#define PRL_ERR_ACCESS_DENINED_TO_RUN_WIZARD PRL_RESULT_DECLARE_ERROR(360)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_TO_CLONE_VM_DEVICE_DENIED Macro

Syntax

```
#define PRL_ERR_ACCESS_TO_CLONE_VM_DEVICE_DENIED PRL_RESULT_DECLARE_ERROR(509)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_TO_VM_DENIED Macro

Syntax

```
#define PRL_ERR_ACCESS_TO_VM_DENIED PRL_RESULT_DECLARE_ERROR(239)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_TO_VM_HDD_DENIED Macro

Syntax

```
#define PRL_ERR_ACCESS_TO_VM_HDD_DENIED PRL_RESULT_DECLARE_ERROR(508)
```

File

PrlErrorsValues.h

PRL_ERR_ACCESS_TOKEN_INVALID Macro

Syntax

```
#define PRL_ERR_ACCESS_TOKEN_INVALID PRL_RESULT_DECLARE_ERROR(60)
```

File

PrlErrorsValues.h

PRL_ERR_ACTION_NOT_SUPPORTED_FOR_CT Macro

Syntax

```
#define PRL_ERR_ACTION_NOT_SUPPORTED_FOR_CT PRL_RESULT_DECLARE_ERROR(44003)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATE_NO_INSTALLED_LICENSE Macro

Syntax

```
#define PRL_ERR_ACTIVATE_NO_INSTALLED_LICENSE PRL_RESULT_DECLARE_ERROR(11704)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATE_TRIAL_LICENSE Macro

Syntax

```
#define PRL_ERR_ACTIVATE_TRIAL_LICENSE PRL_RESULT_DECLARE_ERROR(11705)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE Macro

Syntax

```
#define PRL_ERR_ACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE PRL_RESULT_DECLARE_ERROR(11703)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_COMMON_SERVER_ERROR Macro

Syntax

```
#define PRL_ERR_ACTIVATION_COMMON_SERVER_ERROR PRL_RESULT_DECLARE_ERROR(11718)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_HTTP_REQUEST_FAILED Macro

Syntax

```
#define PRL_ERR_ACTIVATION_HTTP_REQUEST_FAILED PRL_RESULT_DECLARE_ERROR(11713)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_OFFLINE_PERIOD_EXPIRED Macro

Syntax

```
#define PRL_ERR_ACTIVATION_OFFLINE_PERIOD_EXPIRED PRL_RESULT_DECLARE_ERROR(11717)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID Macro

Syntax

```
#define PRL_ERR_ACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID PRL_RESULT_DECLARE_ERROR(11707)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_SERVER_ERROR Macro

Syntax

```
#define PRL_ERR_ACTIVATION_SERVER_ERROR PRL_RESULT_DECLARE_ERROR(11706)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_SERVER_HWIDS_AMOUNT_REACHED Macro

Syntax

```
#define PRL_ERR_ACTIVATION_SERVER_HWIDS_AMOUNT_REACHED PRL_RESULT_DECLARE_ERROR(11710)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_SERVER_KEY_IS_BLACKLISTED Macro

Syntax

```
#define PRL_ERR_ACTIVATION_SERVER_KEY_IS_BLACKLISTED PRL_RESULT_DECLARE_ERROR(11709)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_SERVER_KEY_IS_INVALID Macro

Syntax

```
#define PRL_ERR_ACTIVATION_SERVER_KEY_IS_INVALID PRL_RESULT_DECLARE_ERROR(11708)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_UNABLE_TO_SEND_REQUEST Macro

Syntax

```
#define PRL_ERR_ACTIVATION_UNABLE_TO_SEND_REQUEST PRL_RESULT_DECLARE_ERROR(11712)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_UPDATE_FAILED Macro

Syntax

```
#define PRL_ERR_ACTIVATION_UPDATE_FAILED PRL_RESULT_DECLARE_ERROR(11716)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_FORMAT Macro

- license activation errors

* +++ ALIGNED VALUE TO 11700

Syntax

```
#define PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_FORMAT PRL_RESULT_DECLARE_ERROR(11700)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_MESSAGE Macro

Syntax

```
#define PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_MESSAGE PRL_RESULT_DECLARE_ERROR(11701)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_SIGNATURE Macro

Syntax

```
#define PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_SIGNATURE PRL_RESULT_DECLARE_ERROR(11702)
```

File

PrlErrorsValues.h

PRL_ERR_ACTIVATION_WRONG_SERVER_RESPONSE Macro

Syntax

```
#define PRL_ERR_ACTIVATION_WRONG_SERVER_RESPONSE PRL_RESULT_DECLARE_ERROR(11711)
```

File

PrlErrorsValues.h

PRL_ERR_ADD_HW_FLOPPY_IMAGE_NOT_SPECIFYED Macro

Syntax

```
#define PRL_ERR_ADD_HW_FLOPPY_IMAGE_NOT_SPECIFYED PRL_RESULT_DECLARE_ERROR(317)
```

File

PrlErrorsValues.h

PRL_ERR_ADD_VM_OPERATION_WAS_CANCELED Macro

Syntax

```
#define PRL_ERR_ADD_VM_OPERATION_WAS_CANCELED PRL_RESULT_DECLARE_ERROR(319)
```

File

PrlErrorsValues.h

PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_OPERATION Macro

Syntax

```
#define PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_OPERATION PRL_RESULT_DECLARE_ERROR(41103)
```

File

PrlErrorsValues.h

PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_VM_OPERATION Macro

Syntax

```
#define PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_VM_OPERATION PRL_RESULT_DECLARE_ERROR(41104)
```

File

PrlErrorsValues.h

PRL_ERR_ALIGN_ON_4 Macro

Syntax

```
#define PRL_ERR_ALIGN_ON_4 PRL_RESULT_DECLARE_ERROR(203)
```

File

PrlErrorsValues.h

PRL_ERR_ALREADY_CONNECTED_TO_DISPATCHER Macro

Syntax

```
#define PRL_ERR_ALREADY_CONNECTED_TO_DISPATCHER PRL_RESULT_DECLARE_ERROR(252)
```

File

PrlErrorsValues.h

PRL_ERR_ANOTHER_USER_SESSIONS_PRESENT Macro

Dispatcher service mode errors codes

Syntax

```
#define PRL_ERR_ANOTHER_USER_SESSIONS_PRESENT PRL_RESULT_DECLARE_ERROR(361)
```

File

PrlErrorsValues.h

PRL_ERR_API_INCOMPATIBLE Macro

Syntax

```
#define PRL_ERR_API_INCOMPATIBLE PRL_RESULT_DECLARE_ERROR(422)
```

File

PrlErrorsValues.h

PRL_ERR_API_WASNT_INITIALIZED Macro

Syntax

```
#define PRL_ERR_API_WASNT_INITIALIZED PRL_RESULT_DECLARE_ERROR(287)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_CANNOT_CALCULATE_MD5 Macro

Syntax

```
#define PRL_ERR_APPLIANCE_CANNOT_CALCULATE_MD5 PRL_RESULT_DECLARE_ERROR(41514)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_CANNOT_CHANGE_OWNER Macro

Syntax

```
#define PRL_ERR_APPLIANCE_CANNOT_CHANGE_OWNER PRL_RESULT_DECLARE_ERROR(41510)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_CANNOT_EXTRACT_VM Macro

Syntax

```
#define PRL_ERR_APPLIANCE_CANNOT_EXTRACT_VM PRL_RESULT_DECLARE_ERROR(41508)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_CANNOT_MOVE_VM_BUNDLE Macro

Syntax

```
#define PRL_ERR_APPLIANCE_CANNOT_MOVE_VM_BUNDLE PRL_RESULT_DECLARE_ERROR(41512)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_DIR
Macro

Syntax

```
#define PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_DIR PRL_RESULT_DECLARE_ERROR(41502)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_EXISTS
Macro

Syntax

```
#define PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_EXISTS PRL_RESULT_DECLARE_ERROR(41503)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_DOWNLOAD_PATH_CANNOT_CREATE
Macro

Syntax

```
#define PRL_ERR_APPLIANCE_DOWNLOAD_PATH_CANNOT_CREATE PRL_RESULT_DECLARE_ERROR(41504)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_FOUND
Macro

Syntax

```
#define PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_FOUND PRL_RESULT_DECLARE_ERROR(41505)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_STARTED PRL_RESULT_DECLARE_ERROR(41506)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_EXIT_WITH_ERROR Macro

Syntax

```
#define PRL_ERR_APPLIANCE_EXIT_WITH_ERROR PRL_RESULT_DECLARE_ERROR(41507)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_INSTALL_ALREADY_IN_PROCESS Macro

Syntax

```
#define PRL_ERR_APPLIANCE_INSTALL_ALREADY_IN_PROCESS PRL_RESULT_DECLARE_ERROR(41509)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_INSTALL_WAS_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_APPLIANCE_INSTALL_WAS_NOT_STARTED PRL_RESULT_DECLARE_ERROR(41513)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_INVALID_CONFIG Macro

Install appliance task errors +++ ALIGNED VALUE TO 41500

Syntax

```
#define PRL_ERR_APPLIANCE_INVALID_CONFIG PRL_RESULT_DECLARE_ERROR(41500)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_MISMATCH_MD5 Macro

Syntax

```
#define PRL_ERR_APPLIANCE_MISMATCH_MD5 PRL_RESULT_DECLARE_ERROR(41515)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_NAME_NOT_MATCH_VM_BUNDLE Macro

Syntax

```
#define PRL_ERR_APPLIANCE_NAME_NOT_MATCH_VM_BUNDLE PRL_RESULT_DECLARE_ERROR(41511)
```

File

PrlErrorsValues.h

PRL_ERR_APPLIANCE_USER_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_APPLIANCE_USER_NOT_FOUND PRL_RESULT_DECLARE_ERROR(41501)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_ALREADY_ATTACHED Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_ALREADY_ATTACHED PRL_RESULT_DECLARE_ERROR(54005)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_BUSE_NOT_MOUNTED Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_BUSE_NOT_MOUNTED PRL_RESULT_DECLARE_ERROR(54006)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_CUSTOM_BACKUP_SERVER_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_CUSTOM_BACKUP_SERVER_NOT_SUPPORTED
PRL_RESULT_DECLARE_ERROR(54002)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_FORMAT_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_FORMAT_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(54004)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_INTERNAL_ERROR Macro

Attach backup as a block device errors +++ ALIGNED VALUE TO 54000

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(54000)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_INVALID_STORAGE_URL Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_INVALID_STORAGE_URL PRL_RESULT_DECLARE_ERROR(54001)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_PROTO_ERROR Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_PROTO_ERROR PRL_RESULT_DECLARE_ERROR(54003)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_BACKUP_URL_CHANGE_PROHIBITED Macro

Syntax

```
#define PRL_ERR_ATTACH_BACKUP_URL_CHANGE_PROHIBITED PRL_RESULT_DECLARE_ERROR(54007)
```

File

PrlErrorsValues.h

PRL_ERR_ATTACH_TO_TASK_BAD_SESSION Macro

Syntax

```
#define PRL_ERR_ATTACH_TO_TASK_BAD_SESSION PRL_RESULT_DECLARE_ERROR(349)
```

File

PrlErrorsValues.h

PRL_ERR_AUDIO_ENCODINGS_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_AUDIO_ENCODINGS_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(543)
```

File

PrlErrorsValues.h

PRL_ERR_AUTHENTICATION_FAILED Macro

Syntax

```
#define PRL_ERR_AUTHENTICATION_FAILED PRL_RESULT_DECLARE_ERROR(330)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_ACCESS_TO_VM_DENIED Macro

Syntax

```
#define PRL_ERR_BACKUP_ACCESS_TO_VM_DENIED PRL_RESULT_DECLARE_ERROR(37003)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_ACRONIS_ERR Macro

Syntax

```
#define PRL_ERR_BACKUP_ACRONIS_ERR PRL_RESULT_DECLARE_ERROR(37016)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_BACKUP_CMD_FAILED Macro

Syntax

```
#define PRL_ERR_BACKUP_BACKUP_CMD_FAILED PRL_RESULT_DECLARE_ERROR(37018)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_BACKUP_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_BACKUP_BACKUP_NOT_FOUND PRL_RESULT_DECLARE_ERROR(37011)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_BACKUP_UUID_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_BACKUP_BACKUP_UUID_NOT_FOUND PRL_RESULT_DECLARE_ERROR(37004)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_CANNOT_CREATE_DIRECTORY Macro

Syntax

```
#define PRL_ERR_BACKUP_CANNOT_CREATE_DIRECTORY PRL_RESULT_DECLARE_ERROR(37009)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_CANNOT_REMOVE_DIRECTORY Macro

Syntax

```
#define PRL_ERR_BACKUP_CANNOT_REMOVE_DIRECTORY PRL_RESULT_DECLARE_ERROR(37012)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_CANNOT_SET_PERMISSIONS Macro

Syntax

```
#define PRL_ERR_BACKUP_CANNOT_SET_PERMISSIONS PRL_RESULT_DECLARE_ERROR(37024)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE Macro

Syntax

```
#define PRL_ERR_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE PRL_RESULT_DECLARE_ERROR(37028)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_CREATE_SNAPSHOT_FAILED Macro

Syntax

```
#define PRL_ERR_BACKUP_CREATE_SNAPSHOT_FAILED PRL_RESULT_DECLARE_ERROR(37005)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_CT_ID_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_BACKUP_CT_ID_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(37033)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_DIRECTORY_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_BACKUP_DIRECTORY_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(37008)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_INTERNAL_ERROR Macro

Syntax

```
#define PRL_ERR_BACKUP_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(37001)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_INTERNAL_PROTO_ERROR Macro

VM Backup errors +++ ALIGNED VALUE TO 37000

Syntax

```
#define PRL_ERR_BACKUP_INTERNAL_PROTO_ERROR PRL_RESULT_DECLARE_ERROR(37000)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_LOCKED_FOR_READING Macro

Syntax

```
#define PRL_ERR_BACKUP_LOCKED_FOR_READING PRL_RESULT_DECLARE_ERROR(37013)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_LOCKED_FOR_WRITING Macro

Syntax

```
#define PRL_ERR_BACKUP_LOCKED_FOR_WRITING PRL_RESULT_DECLARE_ERROR(37014)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_REGISTER_VM_FAILED Macro

Syntax

```
#define PRL_ERR_BACKUP_REGISTER_VM_FAILED PRL_RESULT_DECLARE_ERROR(37007)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_REMOVE_PERMISSIONS_DENIED Macro

Syntax

```
#define PRL_ERR_BACKUP_REMOVE_PERMISSIONS_DENIED PRL_RESULT_DECLARE_ERROR(37034)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_REQUIRE_LOGIN_PASSWORD Macro

Syntax

```
#define PRL_ERR_BACKUP_REQUIRE_LOGIN_PASSWORD PRL_RESULT_DECLARE_ERROR(37015)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_CANNOT_CREATE_DIRECTORY Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_CANNOT_CREATE_DIRECTORY PRL_RESULT_DECLARE_ERROR(37041)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_CMD_FAILED Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_CMD_FAILED PRL_RESULT_DECLARE_ERROR(37019)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_DIRECTORY_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_DIRECTORY_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(37040)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_EXISTING_SET_PRIVATE Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_EXISTING_SET_PRIVATE PRL_RESULT_DECLARE_ERROR(37043)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_INTERNAL_ERROR Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(37038)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_INTERNAL_PROTO_ERROR Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_INTERNAL_PROTO_ERROR PRL_RESULT_DECLARE_ERROR(37039)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE PRL_RESULT_DECLARE_ERROR(37027)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_PROHIBIT_WHEN_ATTACHED Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_PROHIBIT_WHEN_ATTACHED PRL_RESULT_DECLARE_ERROR(37042)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_RESTORE_VM_RUNNING Macro

Syntax

```
#define PRL_ERR_BACKUP_RESTORE_VM_RUNNING PRL_RESULT_DECLARE_ERROR(37010)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_SNAPSHOT_OF_PAUSED_VM Macro

Syntax

```
#define PRL_ERR_BACKUP_SNAPSHOT_OF_PAUSED_VM PRL_RESULT_DECLARE_ERROR(37021)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_SWITCH_TO_SNAPSHOT_FAILED Macro

Syntax

```
#define PRL_ERR_BACKUP_SWITCH_TO_SNAPSHOT_FAILED PRL_RESULT_DECLARE_ERROR(37006)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_TIMEOUT_EXCEEDED Macro

Syntax

```
#define PRL_ERR_BACKUP_TIMEOUT_EXCEEDED PRL_RESULT_DECLARE_ERROR(37002)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_TOOL_CANNOT_START Macro

Syntax

```
#define PRL_ERR_BACKUP_TOOL_CANNOT_START PRL_RESULT_DECLARE_ERROR(37032)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_UNSUPPORTED_KEEP_CHAIN Macro

Syntax

```
#define PRL_ERR_BACKUP_UNSUPPORTED_KEEP_CHAIN PRL_RESULT_DECLARE_ERROR(37044)
```

File

PrlErrorsValues.h

PRL_ERR_BACKUP_UNSUPPORTED_STORAGE_VERSION Macro

Syntax

```
#define PRL_ERR_BACKUP_UNSUPPORTED_STORAGE_VERSION PRL_RESULT_DECLARE_ERROR(37045)
```

File

PrlErrorsValues.h

PRL_ERR_BAD_DISP_CONFIG_FILE_SPECIFIED Macro

Syntax

```
#define PRL_ERR_BAD_DISP_CONFIG_FILE_SPECIFIED PRL_RESULT_DECLARE_ERROR(20)
```

File

PrlErrorsValues.h

PRL_ERR_BAD_PARAMETERS Macro

Syntax

```
#define PRL_ERR_BAD_PARAMETERS PRL_RESULT_DECLARE_ERROR(83)
```

File

PrlErrorsValues.h

PRL_ERR_BAD_VM_CONFIG_FILE_SPECIFIED Macro

Syntax

```
#define PRL_ERR_BAD_VM_CONFIG_FILE_SPECIFIED PRL_RESULT_DECLARE_ERROR(32)
```

File

PrlErrorsValues.h

PRL_ERR_BAD_VM_DIR_CONFIG_FILE_SPECIFIED Macro

Syntax

```
#define PRL_ERR_BAD_VM_DIR_CONFIG_FILE_SPECIFIED PRL_RESULT_DECLARE_ERROR(27)
```

File

PrlErrorsValues.h

PRL_ERR_BRIDGE_NOT_FOUND_FOR_NETWORK_ADAPTER Macro

Syntax

```
#define PRL_ERR_BRIDGE_NOT_FOUND_FOR_NETWORK_ADAPTER PRL_RESULT_DECLARE_ERROR(580)
```

File

PrlErrorsValues.h

PRL_ERR_BUFFER_OVERRUN Macro

Syntax

```
#define PRL_ERR_BUFFER_OVERRUN PRL_RESULT_DECLARE_ERROR(6)
```

File

PrlErrorsValues.h

PRL_ERR_BUSE_ENTRY_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_BUSE_ENTRY_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(52003)
```

File

PrlErrorsValues.h

PRL_ERR_BUSE_ENTRY_ALREADY_INITIALIZED Macro

Syntax

```
#define PRL_ERR_BUSE_ENTRY_ALREADY_INITIALIZED PRL_RESULT_DECLARE_ERROR(52004)
```

File

PrlErrorsValues.h

PRL_ERR_BUSE_ENTRY_INVALID Macro

Syntax

```
#define PRL_ERR_BUSE_ENTRY_INVALID PRL_RESULT_DECLARE_ERROR(52002)
```

File

PrlErrorsValues.h

PRL_ERR_BUSE_ENTRY_IO_ERROR Macro

Syntax

```
#define PRL_ERR_BUSE_ENTRY_IO_ERROR PRL_RESULT_DECLARE_ERROR(52005)
```

File

PrlErrorsValues.h

PRL_ERR_BUSE_INTERNAL_ERROR Macro

Syntax

```
#define PRL_ERR_BUSE_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(52001)
```

File

PrlErrorsValues.h

PRL_ERR_BUSE_NOT_MOUNTED Macro

BUSE filesystem errors +++ ALIGNED VALUE TO 52000

Syntax

```
#define PRL_ERR_BUSE_NOT_MOUNTED PRL_RESULT_DECLARE_ERROR(52000)
```

File

PrlErrorsValues.h

PRL_ERR_CAN_NOT_GET_DISK_FREE_SPACE Macro

Syntax

```
#define PRL_ERR_CAN_NOT_GET_DISK_FREE_SPACE PRL_RESULT_DECLARE_ERROR(306)
```

File

PrlErrorsValues.h

PRL_ERR_CANNOT_EDIT_FIREWALL_AT_TRANS_VM_STATE Macro

Syntax

```
#define PRL_ERR_CANNOT_EDIT_FIREWALL_AT_TRANS_VM_STATE PRL_RESULT_DECLARE_ERROR(540)
```

File

PrlErrorsValues.h

PRL_ERR_CANNOT_EDIT_HARDWARE_FOR_PAUSED_VM Macro

Syntax

```
#define PRL_ERR_CANNOT_EDIT_HARDWARE_FOR_PAUSED_VM PRL_RESULT_DECLARE_ERROR(535)
```

File

PrlErrorsValues.h

PRL_ERR_CANNOT_PROCESSING_SAFE_MODE_FOR_INVALID_VM Macro

Syntax

```
#define PRL_ERR_CANNOT_PROCESSING_SAFE_MODE_FOR_INVALID_VM PRL_RESULT_DECLARE_ERROR(445)
```

File

PrlErrorsValues.h

PRL_ERR_CANNOT_PROCESSING_UNDO_DISKS_FOR_INVALID_VM Macro

Syntax

```
#define PRL_ERR_CANNOT_PROCESSING_UNDO_DISKS_FOR_INVALID_VM PRL_RESULT_DECLARE_ERROR(444)
```

File

PrlErrorsValues.h

PRL_ERR_CANNOT_SAVE_REMOTE_DEVICE_STATE Macro

Syntax

```
#define PRL_ERR_CANNOT_SAVE_REMOTE_DEVICE_STATE PRL_RESULT_DECLARE_ERROR(479)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_ALOCA_MEM_FILE_COPY Macro

Syntax

```
#define PRL_ERR_CANT_ALOCA_MEM_FILE_COPY PRL_RESULT_DECLARE_ERROR(109)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_FILE_PERMISSIONS Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_FILE_PERMISSIONS PRL_RESULT_DECLARE_ERROR(339)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_HOST_ID Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_HOST_ID PRL_RESULT_DECLARE_ERROR(561)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_OWNER_OF_DISK_IMAGE_FILE Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_OWNER_OF_DISK_IMAGE_FILE PRL_RESULT_DECLARE_ERROR(394)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_OWNER_OF_FILE Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_OWNER_OF_FILE PRL_RESULT_DECLARE_ERROR(281)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_OWNER_OF_VM_FILE Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_OWNER_OF_VM_FILE PRL_RESULT_DECLARE_ERROR(393)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL PRL_RESULT_DECLARE_ERROR(560)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL_BY_NON_PRIVILEGED_USER Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL_BY_NON_PRIVILEGED_USER
PRL_RESULT_DECLARE_ERROR(562)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN PRL_RESULT_DECLARE_ERROR(574)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN_BY_NON_PRIVILEGED_USER Macro

Syntax

```
#define PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN_BY_NON_PRIVILEGED_USER
PRL_RESULT_DECLARE_ERROR(575)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONFIGURE_PARTITION_HDD Macro

Syntax

```
#define PRL_ERR_CANT_CONFIGURE_PARTITION_HDD PRL_RESULT_DECLARE_ERROR(419)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONFIGURE_PHYSICAL_HDD Macro

Syntax

```
#define PRL_ERR_CANT_CONFIGURE_PHYSICAL_HDD PRL_RESULT_DECLARE_ERROR(338)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONNECT_TO_DISPATCHER Macro

Syntax

```
#define PRL_ERR_CANT_CONNECT_TO_DISPATCHER PRL_RESULT_DECLARE_ERROR(249)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONNECT_TO_DISPATCHER_ITERATIVELY Macro

Syntax

```
#define PRL_ERR_CANT_CONNECT_TO_DISPATCHER_ITERATIVELY PRL_RESULT_DECLARE_ERROR(510)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_CONFIG Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_CONFIG PRL_RESULT_DECLARE_ERROR(45002)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_HDD_IMAGE_FILE Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_HDD_IMAGE_FILE PRL_RESULT_DECLARE_ERROR(310)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_OLD_VM_WITH_INSTALLED_TOOLS Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_OLD_VM_WITH_INSTALLED_TOOLS PRL_RESULT_DECLARE_ERROR(385)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_OTHER_VENDOR_HDD Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_OTHER_VENDOR_HDD PRL_RESULT_DECLARE_ERROR(45004)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_OTHER_VENDOR_VM Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_OTHER_VENDOR_VM PRL_RESULT_DECLARE_ERROR(45003)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_DISKS_PRESENT Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_DISKS_PRESENT PRL_RESULT_DECLARE_ERROR(308)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_SNAPSHOTS_PRESENT Macro

Syntax

```
#define PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_SNAPSHOTS_PRESENT PRL_RESULT_DECLARE_ERROR(309)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CREATE_FLOPPY_IMAGE Macro

Syntax

```
#define PRL_ERR_CANT_CREATE_FLOPPY_IMAGE PRL_RESULT_DECLARE_ERROR(99)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CREATE_HDD_IMAGE Macro

Syntax

```
#define PRL_ERR_CANT_CREATE_HDD_IMAGE PRL_RESULT_DECLARE_ERROR(98)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CREATE_HDD_IMAGE_NO_SPACE Macro

Syntax

```
#define PRL_ERR_CANT_CREATE_HDD_IMAGE_NO_SPACE PRL_RESULT_DECLARE_ERROR(389)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CREATE_PARALLEL_PORT_IMAGE Macro

Syntax

```
#define PRL_ERR_CANT_CREATE_PARALLEL_PORT_IMAGE PRL_RESULT_DECLARE_ERROR(101)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_CREATE_SERIAL_PORT_IMAGE Macro

Syntax

```
#define PRL_ERR_CANT_CREATE_SERIAL_PORT_IMAGE PRL_RESULT_DECLARE_ERROR(400)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_DELETE_FILE Macro

Syntax

```
#define PRL_ERR_CANT_DELETE_FILE PRL_RESULT_DECLARE_ERROR(282)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_EDIT_SUSPENDED_VM Macro

Syntax

```
#define PRL_ERR_CANT_EDIT_SUSPENDED_VM PRL_RESULT_DECLARE_ERROR(104)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_GET_FILE_PERMISSIONS Macro

Syntax

```
#define PRL_ERR_CANT_GET_FILE_PERMISSIONS PRL_RESULT_DECLARE_ERROR(382)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_INIT_REAL_CPUS_INFO Macro

Syntax

```
#define PRL_ERR_CANT_INIT_REAL_CPUS_INFO PRL_RESULT_DECLARE_ERROR(272)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_PARSE_DISP_EVENT Macro

Syntax

```
#define PRL_ERR_CANT_PARSE_DISP_EVENT PRL_RESULT_DECLARE_ERROR(244)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_PARSE_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_CANT_PARSE_VM_CONFIG PRL_RESULT_DECLARE_ERROR(229)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_PREPARE_RECONFIG_DATA Macro

Syntax

```
#define PRL_ERR_CANT_PREPARE_RECONFIG_DATA PRL_RESULT_DECLARE_ERROR(45006)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_RECONFIG_GUEST_OS Macro

Syntax

```
#define PRL_ERR_CANT_RECONFIG_GUEST_OS PRL_RESULT_DECLARE_ERROR(45005)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_RECREATE_HDD Macro

Syntax

```
#define PRL_ERR_CANT_RECREATE_HDD PRL_RESULT_DECLARE_ERROR(329)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_REMOVE_DIR_ENTRY Macro

Syntax

```
#define PRL_ERR_CANT_REMOVE_DIR_ENTRY PRL_RESULT_DECLARE_ERROR(408)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_REMOVE_ENTRY Macro

Syntax

```
#define PRL_ERR_CANT_REMOVE_ENTRY PRL_RESULT_DECLARE_ERROR(82)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_REMOVE_INVALID_VM_AS_NON_ADMIN Macro

Syntax

```
#define PRL_ERR_CANT_REMOVE_INVALID_VM_AS_NON_ADMIN PRL_RESULT_DECLARE_ERROR(383)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_RENAME_DIR_ENTRY Macro

Syntax

```
#define PRL_ERR_CANT_RENAME_DIR_ENTRY PRL_RESULT_DECLARE_ERROR(409)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_RENAME_ENTRY Macro

Syntax

```
#define PRL_ERR_CANT_RENAME_ENTRY PRL_RESULT_DECLARE_ERROR(81)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_REPLACE_FLOPPY_IMAGE Macro

Syntax

```
#define PRL_ERR_CANT_REPLACE_FLOPPY_IMAGE PRL_RESULT_DECLARE_ERROR(333)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_RESIZE_HDD Macro

Syntax

```
#define PRL_ERR_CANT_RESIZE_HDD PRL_RESULT_DECLARE_ERROR(484)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_RESOLVE_HOSTNAME Macro

Syntax

```
#define PRL_ERR_CANT_RESOLVE_HOSTNAME PRL_RESULT_DECLARE_ERROR(511)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_REVERT_VM_SINCE_VTD_DEVICE_ALREADY_USED Macro

Syntax

```
#define PRL_ERR_CANT_REVERT_VM_SINCE_VTD_DEVICE_ALREADY_USED PRL_RESULT_DECLARE_ERROR(469)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_START_VM_SINCE_NO_VTD_DRIVER Macro

Syntax

```
#define PRL_ERR_CANT_START_VM_SINCE_NO_VTD_DRIVER PRL_RESULT_DECLARE_ERROR(461)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_START_VM_SINCE_VTD_DEVICE_ALREADY_USED Macro

Syntax

```
#define PRL_ERR_CANT_START_VM_SINCE_VTD_DEVICE_ALREADY_USED PRL_RESULT_DECLARE_ERROR(463)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_SUSPEND_VM_WITH_BOOTCAMP Macro

Syntax

```
#define PRL_ERR_CANT_SUSPEND_VM_WITH_BOOTCAMP PRL_RESULT_DECLARE_ERROR(20010)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_TO_CHANGE_PERMISSIONS_ON_REMOTE_LOCATION Macro

Syntax

```
#define PRL_ERR_CANT_TO_CHANGE_PERMISSIONS_ON_REMOTE_LOCATION PRL_RESULT_DECLARE_ERROR(417)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_TO_START_VM_TEMPLATE Macro

Syntax

```
#define PRL_ERR_CANT_TO_START_VM_TEMPLATE PRL_RESULT_DECLARE_ERROR(368)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_UNPACK_ARCHIVE Macro

Syntax

```
#define PRL_ERR_CANT_UNPACK_ARCHIVE PRL_RESULT_DECLARE_ERROR(45001)
```

File

PrlErrorsValues.h

PRL_ERR_CANT_UPDATE_DEVICE_INFO Macro

Syntax

```
#define PRL_ERR_CANT_UPDATE_DEVICE_INFO PRL_RESULT_DECLARE_ERROR(332)
```

File

PrlErrorsValues.h

PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT Macro

Syntax

```
#define PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT PRL_RESULT_DECLARE_ERROR(386)
```

File

PrlErrorsValues.h

PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT32 Macro

Syntax

```
#define PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT32 PRL_RESULT_DECLARE_ERROR(387)
```

File

PrlErrorsValues.h

PRL_ERR_CDD_IMAGE_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_CDD_IMAGE_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(208)
```

File

PrlErrorsValues.h

PRL_ERR_CHANGESID_FAILED Macro

Vm changeSID errors +++ ALIGNED VALUE TO 41200

Syntax

```
#define PRL_ERR_CHANGESID_FAILED PRL_RESULT_DECLARE_ERROR(41200)
```

File

PrlErrorsValues.h

PRL_ERR_CHANGESID_GUEST_TOOLS_NOT_AVAILABLE Macro

Syntax

```
#define PRL_ERR_CHANGESID_GUEST_TOOLS_NOT_AVAILABLE PRL_RESULT_DECLARE_ERROR(41201)
```

File

PrlErrorsValues.h

PRL_ERR_CHANGESID_NOT_AVAILABLE Macro

Syntax

```
#define PRL_ERR_CHANGESID_NOT_AVAILABLE PRL_RESULT_DECLARE_ERROR(41204)
```

File

PrlErrorsValues.h

PRL_ERR_CHANGESID_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_CHANGESID_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(41203)
```

File

PrlErrorsValues.h

PRL_ERR_CHANGESID_VM_START_FAILED Macro

Syntax

```
#define PRL_ERR_CHANGESID_VM_START_FAILED PRL_RESULT_DECLARE_ERROR(41202)
```

File

PrlErrorsValues.h

PRL_ERR_CI_CANNOT_COPY_IMAGE_NON_STOPPED_VM Macro

Syntax

```
#define PRL_ERR_CI_CANNOT_COPY_IMAGE_NON_STOPPED_VM PRL_RESULT_DECLARE_ERROR(46701)
```

File

PrlErrorsValues.h

PRL_ERR_CI_DEVICE_IS_NOT_VIRTUAL Macro

Copy device image errors +++ ALIGNED VALUE TO 46700

Syntax

```
#define PRL_ERR_CI_DEVICE_IS_NOT_VIRTUAL PRL_RESULT_DECLARE_ERROR(46700)
```

File

PrlErrorsValues.h

PRL_ERR_CI_PERMISSIONS_DENIED Macro

Syntax

```
#define PRL_ERR_CI_PERMISSIONS_DENIED PRL_RESULT_DECLARE_ERROR(46702)
```

File

PrlErrorsValues.h

PRL_ERR_CLONE_OPERATION_CANCELED Macro

Syntax

```
#define PRL_ERR_CLONE_OPERATION_CANCELED PRL_RESULT_DECLARE_ERROR(107)
```

File

PrlErrorsValues.h

PRL_ERR_CLUSTER_RESOURCE_ERROR Macro

Cluster resource manipulation errors +++ ALIGNED VALUE TO 46900

Syntax

```
#define PRL_ERR_CLUSTER_RESOURCE_ERROR PRL_RESULT_DECLARE_ERROR(46900)
```

File

PrlErrorsValues.h

PRL_ERR_COM_DEVICE_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_COM_DEVICE_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(254)
```

File

PrlErrorsValues.h

PRL_ERR_COM_OUTPUT_FILE_IS_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_COM_OUTPUT_FILE_IS_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(215)
```

File

PrlErrorsValues.h

PRL_ERR_COM_OUTPUT_FILE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_COM_OUTPUT_FILE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(217)
```

File

PrlErrorsValues.h

PRL_ERR_COMMAND_SUPPORTED_ONLY_AT_SERVER_MODE Macro

Syntax

```
#define PRL_ERR_COMMAND_SUPPORTED_ONLY_AT_SERVER_MODE PRL_RESULT_DECLARE_ERROR(421)
```

File

PrlErrorsValues.h

PRL_ERR_COMMON_SERVER_PREFS_BLOCKED_TO_CHANGE Macro

Syntax

```
#define PRL_ERR_COMMON_SERVER_PREFS_BLOCKED_TO_CHANGE PRL_RESULT_DECLARE_ERROR(271)
```

File

PrlErrorsValues.h

PRL_ERR_COMMON_SERVER_PREFS_WERE_CHANGED Macro

Syntax

```
#define PRL_ERR_COMMON_SERVER_PREFS_WERE_CHANGED PRL_RESULT_DECLARE_ERROR(122)
```

File

PrlErrorsValues.h

PRL_ERR_COMPACT_ALREADY_SWITCHED_ON Macro

Syntax

```
#define PRL_ERR_COMPACT_ALREADY_SWITCHED_ON PRL_RESULT_DECLARE_SUCCESS(494)
```

File

PrlErrorsValues.h

PRL_ERR_COMPACT_WRONG_VM_STATE Macro

Syntax

```
#define PRL_ERR_COMPACT_WRONG_VM_STATE PRL_RESULT_DECLARE_ERROR(489)
```

File

PrlErrorsValues.h

PRL_ERR_CONCURRENT_COMMAND_EXECUTING Macro

Syntax

```
#define PRL_ERR_CONCURRENT_COMMAND_EXECUTING PRL_RESULT_DECLARE_ERROR(242)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_OBJECT_UUID Macro

Syntax

```
#define PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_OBJECT_UUID PRL_RESULT_DECLARE_ERROR(346)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_USER_UUID Macro

Syntax

```
#define PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_USER_UUID PRL_RESULT_DECLARE_ERROR(347)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIG_EDIT_COLLISION Macro

Syntax

```
#define PRL_ERR_CONFIG_EDIT_COLLISION PRL_RESULT_DECLARE_ERROR(323)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIG_FILE_NOT_SET Macro

Syntax

```
#define PRL_ERR_CONFIG_FILE_NOT_SET PRL_RESULT_DECLARE_ERROR(38)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIGURE_GENERIC_PCI_TASK_ALREADY_RUN Macro

Syntax

```
#define PRL_ERR_CONFIGURE_GENERIC_PCI_TASK_ALREADY_RUN PRL_RESULT_DECLARE_ERROR(459)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIRMATION_MODE_ALREADY_DISABLED Macro

Syntax

```
#define PRL_ERR_CONFIRMATION_MODE_ALREADY_DISABLED PRL_RESULT_DECLARE_ERROR(41102)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIRMATION_MODE_ALREADY_ENABLED Macro

Syntax

```
#define PRL_ERR_CONFIRMATION_MODE_ALREADY_ENABLED PRL_RESULT_DECLARE_ERROR(41101)
```

File

PrlErrorsValues.h

PRL_ERR_CONFIRMATION_MODE_UNABLE_CHANGE_BY_NOT_ADMIN Macro

confirmation errors +++ ALIGNED VALUE TO 41100

Syntax

```
#define PRL_ERR_CONFIRMATION_MODE_UNABLE_CHANGE_BY_NOT_ADMIN PRL_RESULT_DECLARE_ERROR(41100)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_CLIENT_CERTIFICATE_EXPIRED Macro

Syntax

```
#define PRL_ERR_CONN_CLIENT_CERTIFICATE_EXPIRED PRL_RESULT_DECLARE_ERROR(607)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_CLIENT_CERTIFICATE_INVALID Macro

Syntax

```
#define PRL_ERR_CONN_CLIENT_CERTIFICATE_INVALID PRL_RESULT_DECLARE_ERROR(606)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_CLIENT_CERTIFICATE_REVOKED Macro

Syntax

```
#define PRL_ERR_CONN_CLIENT_CERTIFICATE_REVOKED PRL_RESULT_DECLARE_ERROR(608)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_SERVER_CERTIFICATE_EXPIRED Macro

Syntax

```
#define PRL_ERR_CONN_SERVER_CERTIFICATE_EXPIRED PRL_RESULT_DECLARE_ERROR(610)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_SERVER_CERTIFICATE_INVALID Macro

Syntax

```
#define PRL_ERR_CONN_SERVER_CERTIFICATE_INVALID PRL_RESULT_DECLARE_ERROR(609)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_SERVER_CERTIFICATE_REVOKED Macro

Syntax

```
#define PRL_ERR_CONN_SERVER_CERTIFICATE_REVOKED PRL_RESULT_DECLARE_ERROR(611)
```

File

PrlErrorsValues.h

PRL_ERR_CONN_UNABLE_TO_ESTABLISH_TRUSTED_CHANNEL Macro

Syntax

```
#define PRL_ERR_CONN_UNABLE_TO_ESTABLISH_TRUSTED_CHANNEL PRL_RESULT_DECLARE_ERROR(605)
```

File

PrlErrorsValues.h

PRL_ERR_CONNECT_TO_MOUNTER Macro

Syntax

```
#define PRL_ERR_CONNECT_TO_MOUNTER PRL_RESULT_DECLARE_ERROR(46602)
```

File

PrlErrorsValues.h

PRL_ERR_CONV_HD_CONFLICT Macro

Syntax

```
#define PRL_ERR_CONV_HD_CONFLICT PRL_RESULT_DECLARE_ERROR(41604)
```

File

PrlErrorsValues.h

PRL_ERR_CONV_HD_DISK_TOOL_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_CONV_HD_DISK_TOOL_NOT_STARTED PRL_RESULT_DECLARE_ERROR(41603)
```

File

PrlErrorsValues.h

PRL_ERR_CONV_HD_EXIT_WITH_ERROR Macro

Syntax

```
#define PRL_ERR_CONV_HD_EXIT_WITH_ERROR PRL_RESULT_DECLARE_ERROR(41601)
```

File

PrlErrorsValues.h

PRL_ERR_CONV_HD_NO_ONE_DISK_FOR_CONVERSION Macro

Syntax

```
#define PRL_ERR_CONV_HD_NO_ONE_DISK_FOR_CONVERSION PRL_RESULT_DECLARE_ERROR(41602)
```

File

PrlErrorsValues.h

PRL_ERR_CONV_HD_WRONG_VM_STATE Macro

Convert disks task errors +++ ALIGNED VALUE TO 41600

Syntax

```
#define PRL_ERR_CONV_HD_WRONG_VM_STATE PRL_RESULT_DECLARE_ERROR(41600)
```

File

PrlErrorsValues.h

PRL_ERR_CONVERT_3RD_PARTY_VM_FAILED Macro

Vm convert errors +++ ALIGNED VALUE TO 41300

Syntax

```
#define PRL_ERR_CONVERT_3RD_PARTY_VM_FAILED PRL_RESULT_DECLARE_ERROR(41300)
```

File

PrlErrorsValues.h

PRL_ERR_CONVERT_3RD_PARTY_VM_NO_SPACE Macro

Syntax

```
#define PRL_ERR_CONVERT_3RD_PARTY_VM_NO_SPACE PRL_RESULT_DECLARE_ERROR(41301)
```

File

PrlErrorsValues.h

PRL_ERR_CONVERT_EFI_CONFIG_INVALID Macro

Syntax

```
#define PRL_ERR_CONVERT_EFI_CONFIG_INVALID PRL_RESULT_DECLARE_ERROR(41305)
```

File

PrlErrorsValues.h

PRL_ERR_CONVERT_EFI_GUEST_OS_UNSUPPORTED Macro

Syntax

```
#define PRL_ERR_CONVERT_EFI_GUEST_OS_UNSUPPORTED PRL_RESULT_DECLARE_ERROR(41306)
```

File

PrlErrorsValues.h

PRL_ERR_CONVERT_GUEST_OS_IS_HIBERNATED Macro

Syntax

```
#define PRL_ERR_CONVERT_GUEST_OS_IS_HIBERNATED PRL_RESULT_DECLARE_ERROR(41307)
```

File

PrlErrorsValues.h

PRL_ERR_CONVERT_NO_GUEST_OS_FOUND Macro

Syntax

```
#define PRL_ERR_CONVERT_NO_GUEST_OS_FOUND PRL_RESULT_DECLARE_ERROR(41304)
```

File

PrlErrorsValues.h

PRL_ERR_COPY_CT_TMPL_INTERNAL_ERROR Macro

Syntax

```
#define PRL_ERR_COPY_CT_TMPL_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(31036)
```

File

PrlErrorsValues.h

PRL_ERR_COPY_VM_INFO_FILE Macro

Syntax

```
#define PRL_ERR_COPY_VM_INFO_FILE PRL_RESULT_DECLARE_ERROR(548)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_CANCELLED Macro

Syntax

```
#define PRL_ERR_CORE_STATE_CANCELLED PRL_RESULT_DECLARE_ERROR(20018)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_CHANGE_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_CORE_STATE_CHANGE_VM_CONFIG PRL_RESULT_DECLARE_ERROR(20004)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_CORRUPT_MEM_FILE Macro

Syntax

```
#define PRL_ERR_CORE_STATE_CORRUPT_MEM_FILE PRL_RESULT_DECLARE_ERROR(20003)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_CORRUPT_SAV_FILE Macro

Syntax

```
#define PRL_ERR_CORE_STATE_CORRUPT_SAV_FILE PRL_RESULT_DECLARE_ERROR(20002)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_ERROR_COMMON Macro

- Suspend/Resume codes

+++ ALIGNED VALUE TO 20000

Syntax

```
#define PRL_ERR_CORE_STATE_ERROR_COMMON PRL_RESULT_DECLARE_ERROR(20000)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_INV_SAV_VERSION Macro

Syntax

```
#define PRL_ERR_CORE_STATE_INV_SAV_VERSION PRL_RESULT_DECLARE_ERROR(20001)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_NO_FILE Macro

Syntax

```
#define PRL_ERR_CORE_STATE_NO_FILE PRL_RESULT_DECLARE_ERROR(20017)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_VM_WOULD_RESTART Macro

Syntax

```
#define PRL_ERR_CORE_STATE_VM_WOULD_RESTART PRL_RESULT_DECLARE_ERROR(20019)
```

File

PrlErrorsValues.h

PRL_ERR_CORE_STATE_VM_WOULD_STOP Macro

Syntax

```
#define PRL_ERR_CORE_STATE_VM_WOULD_STOP PRL_RESULT_DECLARE_ERROR(20012)
```

File

PrlErrorsValues.h

PRL_ERR_COULD_NOT_START_FIREWALL_TOOL Macro

Syntax

```
#define PRL_ERR_COULD_NOT_START_FIREWALL_TOOL PRL_RESULT_DECLARE_ERROR(533)
```

File

PrlErrorsValues.h

PRL_ERR_COULDNT_CREATE_AUTHORIZATION_FILE Macro

Syntax

```
#define PRL_ERR_COULDNT_CREATE_AUTHORIZATION_FILE PRL_RESULT_DECLARE_ERROR(295)
```

File

PrlErrorsValues.h

PRL_ERR_COULDNT_SET_PERMISSIONS_TO_AUTHORIZATION_FILE Macro

Syntax

```
#define PRL_ERR_COULDNT_SET_PERMISSIONS_TO_AUTHORIZATION_FILE PRL_RESULT_DECLARE_ERROR(296)
```

File

PrlErrorsValues.h

PRL_ERR_COULDNT_TO_CREATE_HDD_LINKED_CLONE Macro

Syntax

```
#define PRL_ERR_COULDNT_TO_CREATE_HDD_LINKED_CLONE PRL_RESULT_DECLARE_ERROR(514)
```

File

PrlErrorsValues.h

PRL_ERR_CPU_RESTART Macro

Syntax

```
#define PRL_ERR_CPU_RESTART PRL_RESULT_DECLARE_ERROR(380)
```

File

PrlErrorsValues.h

PRL_ERR_CPU_SHUTDOWN Macro

Syntax

```
#define PRL_ERR_CPU_SHUTDOWN PRL_RESULT_DECLARE_ERROR(379)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_DEFAULT_POOL_ARG Macro

Syntax

```
#define PRL_ERR_CPUFEATURES_DEFAULT_POOL_ARG PRL_RESULT_DECLARE_ERROR(54105)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_INCOMPATIBLE_NODE Macro

Syntax

```
#define PRL_ERR_CPUFEATURES_INCOMPATIBLE_NODE PRL_RESULT_DECLARE_ERROR(54106)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_INCORRECT_MASK Macro

Syntax

```
#define PRL_ERR_CPUFEATURES_INCORRECT_MASK PRL_RESULT_DECLARE_ERROR(54101)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_NO_BINARY Macro

CPU features managment errors +++ ALIGNED VALUE TO 54100

Syntax

```
#define PRL_ERR_CPUFEATURES_NO_BINARY PRL_RESULT_DECLARE_ERROR(54100)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_NOT_IN_POOLS Macro

Syntax

```
#define PRL_ERR_CPUFEATURES_NOT_IN_POOLS PRL_RESULT_DECLARE_ERROR(54104)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_POOLS_MANAGMENT Macro

Syntax

```
#define PRL_ERR_CPUFEATURES_POOLS_MANAGMENT PRL_RESULT_DECLARE_ERROR(54102)
```

File

PrlErrorsValues.h

PRL_ERR_CPUFEATURES_RUNNING_VM_OR_CT Macro

Syntax

```
#define PRL_ERR_CPUFEATURES_RUNNING_VM_OR_CT PRL_ERR_RUNNING_VM_OR_CT
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_BOOTABLE_ISO Macro

Unattended ISO errors +++ ALIGNED VALUE TO 40000

Syntax

```
#define PRL_ERR_CREATE_BOOTABLE_ISO PRL_RESULT_DECLARE_ERROR(40000)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_HARD_DISK_WITH_ZERO_SIZE Macro

Syntax

```
#define PRL_ERR_CREATE_HARD_DISK_WITH_ZERO_SIZE PRL_RESULT_DECLARE_ERROR(2003)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_PRIVELEGED_PROCESS Macro

Syntax

```
#define PRL_ERR_CREATE_PRIVELEGED_PROCESS PRL_RESULT_DECLARE_ERROR(17002)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT Macro

Syntax

```
#define PRL_ERR_CREATE_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT PRL_RESULT_DECLARE_ERROR(36005)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_PAUSED Macro

Syntax

```
#define PRL_ERR_CREATE_SNAPSHOT_VM_VTD_PAUSED PRL_RESULT_DECLARE_ERROR(36003)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL
Macro

Syntax

```
#define PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL
PRL_RESULT_DECLARE_ERROR(36009)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL
Macro

Syntax

```
#define PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL
PRL_RESULT_DECLARE_ERROR(36008)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS
Macro

Syntax

```
#define PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS PRL_RESULT_DECLARE_ERROR(36026)
```

File

PrlErrorsValues.h

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL
PRL_RESULT_DECLARE_ERROR(36004)
```

File

PrlErrorsValues.h

PRL_ERR_CT_IS_RUNNING Macro

Syntax

```
#define PRL_ERR_CT_IS_RUNNING PRL_RESULT_DECLARE_ERROR(44008)
```

File

PrlErrorsValues.h

PRL_ERR_CT_MIGRATE_EXTERNAL_PROCESS Macro

Syntax

```
#define PRL_ERR_CT_MIGRATE_EXTERNAL_PROCESS PRL_RESULT_DECLARE_ERROR(31054)
```

File

PrlErrorsValues.h

PRL_ERR_CT_MIGRATE_ID_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_CT_MIGRATE_ID_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(31044)
```

File

PrlErrorsValues.h

PRL_ERR_CT_MIGRATE_INTERNAL_ERROR Macro

Syntax

```
#define PRL_ERR_CT_MIGRATE_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(31035)
```

File

PrlErrorsValues.h

PRL_ERR_CT_MIGRATE_TARGET_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_CT_MIGRATE_TARGET_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(31049)
```

File

PrlErrorsValues.h

PRL_ERR_CT_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_CT_NOT_FOUND PRL_RESULT_DECLARE_ERROR(27027)
```

File

PrlErrorsValues.h

PRL_ERR_CVSRC_IO_ERROR Macro

Captured Video Source errors +++ ALIGNED VALUE TO 46800

Syntax

```
#define PRL_ERR_CVSRC_IO_ERROR PRL_RESULT_DECLARE_ERROR(46800)
```

File

PrlErrorsValues.h

PRL_ERR_CVSRC_NO_CHANNEL Macro

Syntax

```
#define PRL_ERR_CVSRC_NO_CHANNEL PRL_RESULT_DECLARE_ERROR(46802)
```

File

PrlErrorsValues.h

PRL_ERR_CVSRC_NO_OPEN_REQUEST Macro

Syntax

```
#define PRL_ERR_CVSRC_NO_OPEN_REQUEST PRL_RESULT_DECLARE_ERROR(46801)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATE_NO_INSTALLED_LICENSE Macro

Syntax

```
#define PRL_ERR_DEACTIVATE_NO_INSTALLED_LICENSE PRL_RESULT_DECLARE_ERROR(11720)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATE_TRIAL_LICENSE Macro

Syntax

```
#define PRL_ERR_DEACTIVATE_TRIAL_LICENSE PRL_RESULT_DECLARE_ERROR(11721)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE Macro

Syntax

```
#define PRL_ERR_DEACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE PRL_RESULT_DECLARE_ERROR(11719)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_COMMON_SERVER_ERROR Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_COMMON_SERVER_ERROR PRL_RESULT_DECLARE_ERROR(11729)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_HTTP_REQUEST_FAILED Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_HTTP_REQUEST_FAILED PRL_RESULT_DECLARE_ERROR(11728)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_OLD_HWID_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_OLD_HWID_NOT_FOUND PRL_RESULT_DECLARE_ERROR(11730)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID PRL_RESULT_DECLARE_ERROR(11723)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_SERVER_ERROR Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_SERVER_ERROR PRL_RESULT_DECLARE_ERROR(11722)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_SERVER_KEY_IS_BLACKLISTED Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_SERVER_KEY_IS_BLACKLISTED PRL_RESULT_DECLARE_ERROR(11725)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_SERVER_KEY_IS_INVALID Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_SERVER_KEY_IS_INVALID PRL_RESULT_DECLARE_ERROR(11724)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_UNABLE_TO_SEND_REQUEST Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_UNABLE_TO_SEND_REQUEST PRL_RESULT_DECLARE_ERROR(11727)
```

File

PrlErrorsValues.h

PRL_ERR_DEACTIVATION_WRONG_SERVER_RESPONSE Macro

Syntax

```
#define PRL_ERR_DEACTIVATION_WRONG_SERVER_RESPONSE PRL_RESULT_DECLARE_ERROR(11726)
```

File

PrlErrorsValues.h

PRL_ERR_DELETE_UNFINISHED_STATE_FAILED Macro

Syntax

```
#define PRL_ERR_DELETE_UNFINISHED_STATE_FAILED PRL_RESULT_DECLARE_ERROR(577)
```

File

PrlErrorsValues.h

PRL_ERR_DELETING_VM_FROM_CATALOGUE Macro

Syntax

```
#define PRL_ERR_DELETING_VM_FROM_CATALOGUE PRL_RESULT_DECLARE_ERROR(253)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_ALREADY_CONNECTED Macro

- Common Devices Errors

+++ ALIGNED VALUE TO 9000

Syntax

```
#define PRL_ERR_DEV_ALREADY_CONNECTED PRL_RESULT_DECLARE_ERROR(9000)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_ALREADY_DISCONNECTED Macro

Syntax

```
#define PRL_ERR_DEV_ALREADY_DISCONNECTED PRL_RESULT_DECLARE_ERROR(9001)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_DISABLE_AFTER_LOST Macro

Syntax

```
#define PRL_ERR_DEV_DISABLE_AFTER_LOST PRL_RESULT_DECLARE_ERROR(343)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_FLOPPY_CONNECT_FAILED Macro

ALIGNED VALUE TO 1000

Syntax

```
#define PRL_ERR_DEV_FLOPPY_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(1000)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_MAX_CD_EXCEEDED_WITH_DISABLED_CLIENT_SCSII Macro

Syntax

```
#define PRL_ERR_DEV_MAX_CD_EXCEEDED_WITH_DISABLED_CLIENT_SCSII PRL_RESULT_DECLARE_ERROR(10014)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_MAX_CD_HDD_EXCEEDED Macro

Syntax

```
#define PRL_ERR_DEV_MAX_CD_HDD_EXCEEDED PRL_RESULT_DECLARE_ERROR(10001)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_MAX_NUMBER_EXCEEDED Macro

- Other

+++ ALIGNED VALUE TO 10000

Syntax

```
#define PRL_ERR_DEV_MAX_NUMBER_EXCEEDED PRL_RESULT_DECLARE_ERROR(10000)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_PARALLEL_PORT_FILE_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_PARALLEL_PORT_FILE_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(7002)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_PARALLEL_PORT_PHYSICAL_CONNECT_FAILED Macro

- Parallel Port

+++ ALIGNED VALUE TO 7000

Syntax

```
#define PRL_ERR_DEV_PARALLEL_PORT_PHYSICAL_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(7000)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_PARALLEL_PORT_PRINTER_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_PARALLEL_PORT_PRINTER_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(7001)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_PARALLEL_PORT_REMOTE_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_PARALLEL_PORT_REMOTE_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(7003)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_PRINTER_OVERFLOW Macro

Syntax

```
#define PRL_ERR_DEV_PRINTER_OVERFLOW PRL_RESULT_DECLARE_ERROR(9003)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_SERIAL_PORT_FILE_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_SERIAL_PORT_FILE_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(6002)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_SERIAL_PORT_PHYSICAL_CONNECT_FAILED Macro

- Serial Port

+++ ALIGNED VALUE TO 6000

Syntax

```
#define PRL_ERR_DEV_SERIAL_PORT_PHYSICAL_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(6000)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_SERIAL_PORT_PIPE_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_SERIAL_PORT_PIPE_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(6001)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_SERIAL_PORT_REMOTE_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_SERIAL_PORT_REMOTE_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(6003)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_BUSY Macro

Syntax

```
#define PRL_ERR_DEV_USB_BUSY PRL_RESULT_DECLARE_ERROR(8005)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_CHANGEPID Macro

Syntax

```
#define PRL_ERR_DEV_USB_CHANGEPID PRL_RESULT_DECLARE_ERROR(8007)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED Macro

Syntax

```
#define PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED PRL_RESULT_DECLARE_ERROR(8003)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED2 Macro

Syntax

```
#define PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED2 PRL_RESULT_DECLARE_ERROR(8004)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_INSTALL_DRIVER_FAILED Macro

Syntax

```
#define PRL_ERR_DEV_USB_INSTALL_DRIVER_FAILED PRL_RESULT_DECLARE_ERROR(8001)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_NO_FREE_PORTS Macro

Syntax

```
#define PRL_ERR_DEV_USB_NO_FREE_PORTS PRL_RESULT_DECLARE_ERROR(8002)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_NOT_CONFIGURED Macro

Syntax

```
#define PRL_ERR_DEV_USB_NOT_CONFIGURED PRL_RESULT_DECLARE_ERROR(8008)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_OPEN_MANAGER_FAILED Macro

- USB

+++ ALIGNED VALUE TO 8000

Syntax

```
#define PRL_ERR_DEV_USB_OPEN_MANAGER_FAILED PRL_RESULT_DECLARE_ERROR(8000)
```

File

PrlErrorsValues.h

PRL_ERR_DEV_USB_REUSE Macro

Syntax

```
#define PRL_ERR_DEV_USB_REUSE PRL_RESULT_DECLARE_ERROR(8006)
```

File

PrlErrorsValues.h

PRL_ERR_DIRECTORY_DOES_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_DIRECTORY_DOES_NOT_EXIST PRL_RESULT_DECLARE_ERROR(75)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_BLOCK_CREATED Macro

Syntax

```
#define PRL_ERR_DISK_BLOCK_CREATED PRL_RESULT_DECLARE_ERROR(21045)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_BLOCK_PARTIALLY_PROCESSED Macro

Syntax

```
#define PRL_ERR_DISK_BLOCK_PARTIALLY_PROCESSED PRL_RESULT_DECLARE_ERROR(21019)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_BLOCK_SEARCH_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_BLOCK_SEARCH_FAILED PRL_RESULT_DECLARE_ERROR(22003)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_BLOCK_SKIPPED Macro

Syntax

```
#define PRL_ERR_DISK_BLOCK_SKIPPED PRL_RESULT_DECLARE_ERROR(21018)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_BOOTCAMP_WRITE_MBR Macro

Syntax

```
#define PRL_ERR_DISK_BOOTCAMP_WRITE_MBR PRL_RESULT_DECLARE_ERROR(21059)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_CANT_INITIALIZE_IMAGE Macro

Syntax

```
#define PRL_ERR_DISK_CANT_INITIALIZE_IMAGE PRL_RESULT_DECLARE_ERROR(21049)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_COMPRESSED_FILE_EMPTY Macro

Syntax

```
#define PRL_ERR_DISK_COMPRESSED_FILE_EMPTY PRL_RESULT_DECLARE_ERROR(22001)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_CREATE_IMAGE_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_CREATE_IMAGE_ERROR PRL_RESULT_DECLARE_ERROR(21016)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_DATA_NOT_FOUND Macro

- User parameters errors

* +++ ALIGNED VALUE TO 23000

Syntax

```
#define PRL_ERR_DISK_DATA_NOT_FOUND PRL_RESULT_DECLARE_ERROR(23000)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_DIR_CREATE_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_DIR_CREATE_ERROR PRL_RESULT_DECLARE_ERROR(21015)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_DISK_NOT_OPENED Macro

Syntax

```
#define PRL_ERR_DISK_DISK_NOT_OPENED PRL_RESULT_DECLARE_ERROR(21021)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_ENLARGE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_ENLARGE_FAILED PRL_RESULT_DECLARE_ERROR(21057)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_FAT32_SIZE_EXCEEDED Macro

Syntax

```
#define PRL_ERR_DISK_FAT32_SIZE_EXCEEDED PRL_RESULT_DECLARE_ERROR(21023)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_FILE_CREATE_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_FILE_CREATE_ERROR PRL_RESULT_DECLARE_ERROR(21013)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_FILE_EXISTS Macro

Syntax

```
#define PRL_ERR_DISK_FILE_EXISTS PRL_RESULT_DECLARE_ERROR(21012)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_FILE_OPEN_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_FILE_OPEN_ERROR PRL_RESULT_DECLARE_ERROR(21014)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_FSYNC_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_FSYNC_FAILED PRL_RESULT_DECLARE_ERROR(21031)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_FULLFSYNC_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_FULLFSYNC_FAILED PRL_RESULT_DECLARE_ERROR(21032)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_GENERIC_ERROR Macro

- Range reserved for different disk errors

+++ ALIGNED VALUE TO 21000

Syntax

```
#define PRL_ERR_DISK_GENERIC_ERROR PRL_RESULT_DECLARE_ERROR(21000)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_GET_MOUNTPATH_FAILED Macro

- Bootcamp errors

* +++ ALIGNED VALUE TO 24000

Syntax

```
#define PRL_ERR_DISK_GET_MOUNTPATH_FAILED PRL_RESULT_DECLARE_ERROR(24000)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_GET_PARAMS_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_GET_PARAMS_FAILED PRL_RESULT_DECLARE_ERROR(24003)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_GET_SIZE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_GET_SIZE_FAILED PRL_RESULT_DECLARE_ERROR(21064)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_GPT_MBR_NOT_EQUAL Macro

Syntax

```
#define PRL_ERR_DISK_GPT_MBR_NOT_EQUAL PRL_RESULT_DECLARE_ERROR(21065)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_GROUP_INTERSECT Macro

Syntax

```
#define PRL_ERR_DISK_GROUP_INTERSECT PRL_RESULT_DECLARE_ERROR(21046)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_IDENTIFY_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_IDENTIFY_FAILED PRL_RESULT_DECLARE_ERROR(24004)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_IMAGE_BUSY Macro

Syntax

```
#define PRL_ERR_DISK_IMAGE_BUSY PRL_RESULT_DECLARE_ERROR(21044)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_IMAGES_CORRUPTED Macro

Syntax

```
#define PRL_ERR_DISK_IMAGES_CORRUPTED PRL_RESULT_DECLARE_ERROR(21008)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_IMPERSONATE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_IMPERSONATE_FAILED PRL_RESULT_DECLARE_ERROR(21060)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_INCORRECTLY_CLOSED Macro

Syntax

```
#define PRL_ERR_DISK_INCORRECTLY_CLOSED PRL_RESULT_DECLARE_ERROR(21047)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_INSUFFICIENT_SPACE Macro

Syntax

```
#define PRL_ERR_DISK_INSUFFICIENT_SPACE PRL_RESULT_DECLARE_ERROR(21022)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_INTERNAL_CLASS_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_INTERNAL_CLASS_ERROR PRL_RESULT_DECLARE_ERROR(21025)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_INVALID_BLOCKSIZE Macro

Syntax

```
#define PRL_ERR_DISK_INVALID_BLOCKSIZE PRL_RESULT_DECLARE_ERROR(21009)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_INVALID_FORMAT Macro

Syntax

```
#define PRL_ERR_DISK_INVALID_FORMAT PRL_RESULT_DECLARE_ERROR(21033)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_INVALID_PARAMETERS Macro

Syntax

```
#define PRL_ERR_DISK_INVALID_PARAMETERS PRL_RESULT_DECLARE_ERROR(21011)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_MEMORY_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_MEMORY_ERROR PRL_RESULT_DECLARE_ERROR(21020)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_MOUNT_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_MOUNT_FAILED PRL_RESULT_DECLARE_ERROR(24001)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_MOUNTED_OVERLAP Macro

Syntax

```
#define PRL_ERR_DISK_MOUNTED_OVERLAP PRL_RESULT_DECLARE_ERROR(21063)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_NOT_IMPLEMENTED Macro

Syntax

```
#define PRL_ERR_DISK_NOT_IMPLEMENTED PRL_RESULT_DECLARE_ERROR(21042)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_NOT_PERMITTED Macro

Syntax

```
#define PRL_ERR_DISK_NOT_PERMITTED PRL_RESULT_DECLARE_ERROR(21024)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_NOT_VALID_OFFSET Macro

Syntax

```
#define PRL_ERR_DISK_NOT_VALID_OFFSET PRL_RESULT_DECLARE_ERROR(21056)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_NULL_PART_SIZE Macro

Syntax

```
#define PRL_ERR_DISK_NULL_PART_SIZE PRL_RESULT_DECLARE_ERROR(21051)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_OFFSETS_FIXED Macro

Syntax

```
#define PRL_ERR_DISK_OFFSETS_FIXED PRL_RESULT_DECLARE_ERROR(21048)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_OPERATION_ABORTED Macro

Syntax

```
#define PRL_ERR_DISK_OPERATION_ABORTED PRL_RESULT_DECLARE_ERROR(21036)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_OPERATION_IN_PROGRESS Macro

Syntax

```
#define PRL_ERR_DISK_OPERATION_IN_PROGRESS PRL_RESULT_DECLARE_ERROR(21017)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_OPERATION_NOT_ALLOWED Macro

Syntax

```
#define PRL_ERR_DISK_OPERATION_NOT_ALLOWED PRL_RESULT_DECLARE_ERROR(21035)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_PARTITION_INVALID_NAME Macro

Syntax

```
#define PRL_ERR_DISK_PARTITION_INVALID_NAME PRL_RESULT_DECLARE_ERROR(21054)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_PARTITION_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_DISK_PARTITION_NOT_FOUND PRL_RESULT_DECLARE_ERROR(21053)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_PARTITIONS_TABLE_CYCLE Macro

Syntax

```
#define PRL_ERR_DISK_PARTITIONS_TABLE_CYCLE PRL_RESULT_DECLARE_ERROR(21043)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_POINTERS_MIXED_UP Macro

Syntax

```
#define PRL_ERR_DISK_POINTERS_MIXED_UP PRL_RESULT_DECLARE_ERROR(22004)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_POSSIBLE_OVERRUN Macro

Syntax

```
#define PRL_ERR_DISK_POSSIBLE_OVERRUN PRL_RESULT_DECLARE_ERROR(21058)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_READ_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_READ_FAILED PRL_RESULT_DECLARE_ERROR(21029)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_READ_OUT_DISK Macro

Syntax

```
#define PRL_ERR_DISK_READ_OUT_DISK PRL_RESULT_DECLARE_ERROR(21028)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_RENAME_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_RENAME_ERROR PRL_RESULT_DECLARE_ERROR(21034)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_RESERVED_WORD Macro

Syntax

```
#define PRL_ERR_DISK_RESERVED_WORD PRL_RESULT_DECLARE_ERROR(23001)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_RESIZE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_RESIZE_FAILED PRL_RESULT_DECLARE_ERROR(41003)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_RESIZE_SIZE_TOO_LOW Macro

Syntax

```
#define PRL_ERR_DISK_RESIZE_SIZE_TOO_LOW PRL_RESULT_DECLARE_ERROR(41002)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_RESIZE_WITH_SNAPSHOTS_NOT_ALLOWED Macro

Syntax

```
#define PRL_ERR_DISK_RESIZE_WITH_SNAPSHOTS_NOT_ALLOWED PRL_RESULT_DECLARE_ERROR(41001)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_RESIZER_NOT_FOUND Macro

VM image resizer errors +++ ALIGNED VALUE TO 41000

Syntax

```
#define PRL_ERR_DISK_RESIZER_NOT_FOUND PRL_RESULT_DECLARE_ERROR(41000)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_SET_CACHING_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_SET_CACHING_FAILED PRL_RESULT_DECLARE_ERROR(21030)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_SET_FILE_SIZE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_SET_FILE_SIZE_FAILED PRL_RESULT_DECLARE_ERROR(22005)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_SHARED_BLOCK Macro

- Compressing errors

* +++ ALIGNED VALUE TO 22000

Syntax

```
#define PRL_ERR_DISK_SHARED_BLOCK PRL_RESULT_DECLARE_ERROR(22000)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_SHARING_VIOLATION Macro

Syntax

```
#define PRL_ERR_DISK_SHARING_VIOLATION PRL_RESULT_DECLARE_ERROR(21037)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_SMALL_IMAGE_SIZE Macro

Syntax

```
#define PRL_ERR_DISK_SMALL_IMAGE_SIZE PRL_RESULT_DECLARE_ERROR(21050)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_SNAPSHOTS_CORRUPTED Macro

Syntax

```
#define PRL_ERR_DISK_SNAPSHOTS_CORRUPTED PRL_RESULT_DECLARE_ERROR(21006)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_STATES_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_STATES_ERROR PRL_RESULT_DECLARE_ERROR(21010)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_STORAGE_CORRUPTED Macro

Syntax

```
#define PRL_ERR_DISK_STORAGE_CORRUPTED PRL_RESULT_DECLARE_ERROR(21007)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_TOOL_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_DISK_TOOL_NOT_FOUND PRL_RESULT_DECLARE_ERROR(492)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_TOOL_PROCESS_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_TOOL_PROCESS_ERROR PRL_RESULT_DECLARE_ERROR(491)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_TRUNCATE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_TRUNCATE_FAILED PRL_RESULT_DECLARE_ERROR(22002)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_UNALIGNED Macro

Syntax

```
#define PRL_ERR_DISK_UNALIGNED PRL_RESULT_DECLARE_ERROR(21055)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_UNCOMMITED_OPERATION Macro

Syntax

```
#define PRL_ERR_DISK_UNCOMMITED_OPERATION PRL_RESULT_DECLARE_ERROR(21062)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_UNMOUNT_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_UNMOUNT_FAILED PRL_RESULT_DECLARE_ERROR(24002)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_USER_INTERRUPTED Macro

Syntax

```
#define PRL_ERR_DISK_USER_INTERRUPTED PRL_RESULT_DECLARE_ERROR(21038)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_WRITE_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_WRITE_FAILED PRL_RESULT_DECLARE_ERROR(21027)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_WRITE_OUT_DISK Macro

Syntax

```
#define PRL_ERR_DISK_WRITE_OUT_DISK PRL_RESULT_DECLARE_ERROR(21026)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_WRITE_REAL_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_WRITE_REAL_FAILED PRL_RESULT_DECLARE_ERROR(21041)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_DIFFERS_FROM_REAL Macro

Syntax

```
#define PRL_ERR_DISK_XML_DIFFERS_FROM_REAL PRL_RESULT_DECLARE_ERROR(21066)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_INVALID Macro

Syntax

```
#define PRL_ERR_DISK_XML_INVALID PRL_RESULT_DECLARE_ERROR(21002)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_INVALID_VERSION Macro

Syntax

```
#define PRL_ERR_DISK_XML_INVALID_VERSION PRL_RESULT_DECLARE_ERROR(21003)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_LARGE_FILE Macro

Syntax

```
#define PRL_ERR_DISK_XML_LARGE_FILE PRL_RESULT_DECLARE_ERROR(21005)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_LOCKED Macro

Syntax

```
#define PRL_ERR_DISK_XML_LOCKED PRL_RESULT_DECLARE_ERROR(21052)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_MISSING Macro

Syntax

```
#define PRL_ERR_DISK_XML_MISSING PRL_RESULT_DECLARE_ERROR(21061)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_OPEN_FAILED Macro

Syntax

```
#define PRL_ERR_DISK_XML_OPEN_FAILED PRL_RESULT_DECLARE_ERROR(21001)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_PARTITION_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_DISK_XML_PARTITION_NOT_FOUND PRL_RESULT_DECLARE_ERROR(21067)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_SAVE_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_XML_SAVE_ERROR PRL_RESULT_DECLARE_ERROR(21004)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_SAVE_REMOVE_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_XML_SAVE_REMOVE_ERROR PRL_RESULT_DECLARE_ERROR(21039)
```

File

PrlErrorsValues.h

PRL_ERR_DISK_XML_SAVE_RENAME_ERROR Macro

Syntax

```
#define PRL_ERR_DISK_XML_SAVE_RENAME_ERROR PRL_RESULT_DECLARE_ERROR(21040)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_AUTO_COMPRESS_PERIOD_OUT_OF_RANGE Macro

Syntax

```
#define PRL_ERR_DISP_AUTO_COMPRESS_PERIOD_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(488)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_CANNOT_COMPACT_VM_HARD_DISK Macro

Syntax

```
#define PRL_ERR_DISP_CANNOT_COMPACT_VM_HARD_DISK PRL_RESULT_DECLARE_ERROR(493)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_CONFIG_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_DISP_CONFIG_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(25)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_CONFIG_FILE_NOT_SET Macro

Syntax

```
#define PRL_ERR_DISP_CONFIG_FILE_NOT_SET PRL_RESULT_DECLARE_ERROR(40)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_CONFIG_WRITE_ERR Macro

Syntax

```
#define PRL_ERR_DISP_CONFIG_WRITE_ERR PRL_RESULT_DECLARE_ERROR(66)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_LOGON_ACTIONS_REACHED_UP_LIMIT Macro

Syntax

```
#define PRL_ERR_DISP_LOGON_ACTIONS_REACHED_UP_LIMIT PRL_RESULT_DECLARE_ERROR(415)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_SHUTDOWN_IN_PROCESS Macro

Syntax

```
#define PRL_ERR_DISP_SHUTDOWN_IN_PROCESS PRL_RESULT_DECLARE_ERROR(404)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_START_VM_FOR_COMPACT_FAILED Macro

Syntax

```
#define PRL_ERR_DISP_START_VM_FOR_COMPACT_FAILED PRL_RESULT_DECLARE_ERROR(486)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_TIME_MACHINE_IS_RUNNING Macro

Syntax

```
#define PRL_ERR_DISP_TIME_MACHINE_IS_RUNNING PRL_RESULT_DECLARE_ERROR(504)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_VM_COMMAND_CANT_BE_EXECUTED Macro

Syntax

```
#define PRL_ERR_DISP_VM_COMMAND_CANT_BE_EXECUTED PRL_RESULT_DECLARE_ERROR(416)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_VM_IS_NOT_STARTED Macro

Dispatcher errors codes

Syntax

```
#define PRL_ERR_DISP_VM_IS_NOT_STARTED PRL_RESULT_DECLARE_ERROR(377)
```

File

PrlErrorsValues.h

PRL_ERR_DISP_VM_IS_NOT_STOPPED Macro

Syntax

```
#define PRL_ERR_DISP_VM_IS_NOT_STOPPED PRL_RESULT_DECLARE_ERROR(557)
```

File

PrlErrorsValues.h

PRL_ERR_DISP2DISP_SESSION_ALREADY_AUTHORIZED Macro

Dispatcher-dispatcher protocol errors codes +++ ALIGNED VALUE TO 30000

Syntax

```
#define PRL_ERR_DISP2DISP_SESSION_ALREADY_AUTHORIZED PRL_RESULT_DECLARE_ERROR(30000)
```

File

PrlErrorsValues.h

PRL_ERR_DISP2DISP_WRONG_USER_SESSION_UUID Macro

Syntax

```
#define PRL_ERR_DISP2DISP_WRONG_USER_SESSION_UUID PRL_RESULT_DECLARE_ERROR(30001)
```

File

PrlErrorsValues.h

PRL_ERR_DISPATCHER_SERVICE_MODE Macro

Syntax

```
#define PRL_ERR_DISPATCHER_SERVICE_MODE PRL_RESULT_DECLARE_ERROR(364)
```

File

PrlErrorsValues.h

PRL_ERR_DISPLAY_ENCODINGS_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_DISPLAY_ENCODINGS_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(526)
```

File

PrlErrorsValues.h

PRL_ERR_DISPLAY_FORMAT_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_DISPLAY_FORMAT_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(549)
```

File

PrlErrorsValues.h

PRL_ERR_DOUBLE_INIT Macro

Syntax

```
#define PRL_ERR_DOUBLE_INIT PRL_RESULT_DECLARE_ERROR(11)
```

File

PrlErrorsValues.h

PRL_ERR_DROP_SUSPEND_FOR_HDD_QUEST Macro

Syntax

```
#define PRL_ERR_DROP_SUSPEND_FOR_HDD_QUEST PRL_RESULT_DECLARE_ERROR(405)
```

File

PrlErrorsValues.h

PRL_ERR_ENCRYPTION_COMMIT_PROHIBITED Macro

Syntax

```
#define PRL_ERR_ENCRYPTION_COMMIT_PROHIBITED PRL_RESULT_DECLARE_ERROR(54201)
```

File

PrlErrorsValues.h

PRL_ERR_ENCRYPTION_COMMIT_REJECTED Macro

Disk encryption errors +++ ALIGNED VALUE TO 54200

Syntax

```
#define PRL_ERR_ENCRYPTION_COMMIT_REJECTED PRL_RESULT_DECLARE_ERROR(54200)
```

File

PrlErrorsValues.h

PRL_ERR_ENTRY_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_ENTRY_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(80)
```

File

PrlErrorsValues.h

PRL_ERR_ENTRY_DIR_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_ENTRY_DIR_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(410)
```

File

PrlErrorsValues.h

PRL_ERR_ENTRY_DIR_DOES_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_ENTRY_DIR_DOES_NOT_EXIST PRL_RESULT_DECLARE_ERROR(411)
```

File

PrlErrorsValues.h

PRL_ERR_ENTRY_DOES_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_ENTRY_DOES_NOT_EXIST PRL_RESULT_DECLARE_ERROR(79)
```

File

PrlErrorsValues.h

PRL_ERR_EXCEED_LIMIT_MAX_RUNNING_VMS Macro

Syntax

```
#define PRL_ERR_EXCEED_LIMIT_MAX_RUNNING_VMS PRL_RESULT_DECLARE_ERROR(388)
```

File

PrlErrorsValues.h

PRL_ERR_EXCEED_MEMORY_LIMIT Macro

Syntax

```
#define PRL_ERR_EXCEED_MEMORY_LIMIT PRL_RESULT_DECLARE_ERROR(205)
```

File

PrlErrorsValues.h

PRL_ERR_EXT_DISK_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_EXT_DISK_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(578)
```

File

PrlErrorsValues.h

PRL_ERR_EXT_DISK_CANT_RENAME Macro

Syntax

```
#define PRL_ERR_EXT_DISK_CANT_RENAME PRL_RESULT_DECLARE_ERROR(579)
```

File

PrlErrorsValues.h

PRL_ERR_FAILED_TO_AUTH_ON_BACKUP_SERVER Macro

Syntax

```
#define PRL_ERR_FAILED_TO_AUTH_ON_BACKUP_SERVER PRL_RESULT_DECLARE_ERROR(37023)
```

File

PrlErrorsValues.h

PRL_ERR_FAILED_TO_CONNECT_TO_BACKUP_SERVER Macro

Syntax

```
#define PRL_ERR_FAILED_TO_CONNECT_TO_BACKUP_SERVER PRL_RESULT_DECLARE_ERROR(37022)
```

File

PrlErrorsValues.h

PRL_ERR_FAILED_TO_START_VNC_SERVER Macro

Syntax

```
#define PRL_ERR_FAILED_TO_START_VNC_SERVER PRL_RESULT_DECLARE_ERROR(35003)
```

File

PrlErrorsValues.h

PRL_ERR_FAILED_TO_STOP_VNC_SERVER Macro

Syntax

```
#define PRL_ERR_FAILED_TO_STOP_VNC_SERVER PRL_RESULT_DECLARE_ERROR(35004)
```

File

PrlErrorsValues.h

PRL_ERR_FAILURE Macro

Syntax

```
#define PRL_ERR_FAILURE PRL_RESULT_DECLARE_ERROR(9)
```

File

PrlErrorsValues.h

PRL_ERR_FAILURE_ON_VM_DESTROYED Macro

Syntax

```
#define PRL_ERR_FAILURE_ON_VM_DESTROYED PRL_RESULT_DECLARE_ERROR(462)
```

File

PrlErrorsValues.h

PRL_ERR_FDD_IMAGE_CLONE_TO_SELF Macro

Syntax

```
#define PRL_ERR_FDD_IMAGE_CLONE_TO_SELF PRL_RESULT_DECLARE_ERROR(315)
```

File

PrlErrorsValues.h

PRL_ERR_FDD_IMAGE_COPY Macro

Syntax

```
#define PRL_ERR_FDD_IMAGE_COPY PRL_RESULT_DECLARE_ERROR(314)
```

File

PrlErrorsValues.h

PRL_ERR_FDD_IMAGE_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_FDD_IMAGE_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(209)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_DISK_SPACE_ERROR Macro

Syntax

```
#define PRL_ERR_FILE_DISK_SPACE_ERROR PRL_RESULT_DECLARE_ERROR(373)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_FILE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(322)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_FILE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(10)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_OR_DIR_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_FILE_OR_DIR_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(355)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_READ_ERROR Macro

Syntax

```
#define PRL_ERR_FILE_READ_ERROR PRL_RESULT_DECLARE_ERROR(110)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_CANT_CREATE_DST_FILE Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_CANT_CREATE_DST_FILE PRL_RESULT_DECLARE_ERROR(26004)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_CANT_LOCATE_SRC_FILE Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_CANT_LOCATE_SRC_FILE PRL_RESULT_DECLARE_ERROR(26001)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_CANT_READ_SRC_FILE Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_CANT_READ_SRC_FILE PRL_RESULT_DECLARE_ERROR(26002)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_CANT_WRITE_DATA_TO_DST_FILE Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_CANT_WRITE_DATA_TO_DST_FILE PRL_RESULT_DECLARE_ERROR(26005)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_CLIENT_NOT_CONNECTED Macro

- File Transfer errors

+++ ALIGNED VALUE TO 26000

Syntax

```
#define PRL_ERR_FILE_TRANSFER_CLIENT_NOT_CONNECTED PRL_RESULT_DECLARE_ERROR(26000)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_DST_FILE_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_DST_FILE_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(26006)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_INVALID_ARGUMENTS Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_INVALID_ARGUMENTS PRL_RESULT_DECLARE_ERROR(26009)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_INVALID_CREDENTIALS Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_INVALID_CREDENTIALS PRL_RESULT_DECLARE_ERROR(26003)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_OPERATION_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_OPERATION_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(26007)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_TRANSFER_UPLOAD_CANCELED_BY_USER Macro

Syntax

```
#define PRL_ERR_FILE_TRANSFER_UPLOAD_CANCELED_BY_USER PRL_RESULT_DECLARE_ERROR(26008)
```

File

PrlErrorsValues.h

PRL_ERR_FILE_WRITE_ERROR Macro

Syntax

```
#define PRL_ERR_FILE_WRITE_ERROR PRL_RESULT_DECLARE_ERROR(372)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_CANT_CREATE_DIR Macro

Syntax

```
#define PRL_ERR_FILECOPY_CANT_CREATE_DIR PRL_RESULT_DECLARE_ERROR(31021)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_CANT_OPEN_FILE Macro

Syntax

```
#define PRL_ERR_FILECOPY_CANT_OPEN_FILE PRL_RESULT_DECLARE_ERROR(31022)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_CANT_WRITE Macro

Syntax

```
#define PRL_ERR_FILECOPY_CANT_WRITE PRL_RESULT_DECLARE_ERROR(31023)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_DIR_EXIST Macro

Syntax

```
#define PRL_ERR_FILECOPY_DIR_EXIST PRL_RESULT_DECLARE_ERROR(31019)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_FILE_EXIST Macro

Syntax

```
#define PRL_ERR_FILECOPY_FILE_EXIST PRL_RESULT_DECLARE_ERROR(31020)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_INTERNAL Macro

Syntax

```
#define PRL_ERR_FILECOPY_INTERNAL PRL_RESULT_DECLARE_ERROR(31024)
```

File

PrlErrorsValues.h

PRL_ERR_FILECOPY_PROTOCOL Macro

Syntax

```
#define PRL_ERR_FILECOPY_PROTOCOL PRL_RESULT_DECLARE_ERROR(31018)
```

File

PrlErrorsValues.h

PRL_ERR_FIREWALL_TOOL_EXECUTED_WITH_ERROR Macro

Syntax

```
#define PRL_ERR_FIREWALL_TOOL_EXECUTED_WITH_ERROR PRL_RESULT_DECLARE_ERROR(534)
```

File

PrlErrorsValues.h

PRL_ERR_FIXME Macro

Syntax

```
#define PRL_ERR_FIXME PRL_RESULT_DECLARE_ERROR(227)
```

File

PrlErrorsValues.h

PRL_ERR_FLOPPY_DRIVE_INVALID Macro

Syntax

```
#define PRL_ERR_FLOPPY_DRIVE_INVALID PRL_RESULT_DECLARE_ERROR(226)
```

File

PrlErrorsValues.h

PRL_ERR_FLOPPY_IMAGE_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_FLOPPY_IMAGE_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(256)
```

File

PrlErrorsValues.h

PRL_ERR_FLOPPY_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_FLOPPY_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(95)
```

File

PrlErrorsValues.h

PRL_ERR_FORCE_REG_ON_SHARED_STORAGE_IS_NEEDED Macro

Syntax

```
#define PRL_ERR_FORCE_REG_ON_SHARED_STORAGE_IS_NEEDED PRL_ERR_REG_PSTORAGE_REVOKE_IS_NEEDS
```

File

PrlErrorsValues.h

PRL_ERR_FREE_DISC_SPACE_FOR_CLONE Macro

Syntax

```
#define PRL_ERR_FREE_DISC_SPACE_FOR_CLONE PRL_RESULT_DECLARE_ERROR(371)
```

File

PrlErrorsValues.h

PRL_ERR_FREE_DISK_SPACE_FOR_CREATE_SNAPSHOT Macro

Syntax

```
#define PRL_ERR_FREE_DISK_SPACE_FOR_CREATE_SNAPSHOT PRL_RESULT_DECLARE_ERROR(442)
```

File

PrlErrorsValues.h

PRL_ERR_FREE_DISK_SPACE_FOR_REVERT_TO_SNAPSHOT Macro

Syntax

```
#define PRL_ERR_FREE_DISK_SPACE_FOR_REVERT_TO_SNAPSHOT PRL_RESULT_DECLARE_ERROR(443)
```

File

PrlErrorsValues.h

PRL_ERR_GET_DISK_FREE_SPACE_FAILED Macro

Syntax

```
#define PRL_ERR_GET_DISK_FREE_SPACE_FAILED PRL_RESULT_DECLARE_ERROR(556)
```

File

PrlErrorsValues.h

PRL_ERR_GET_LICENSE_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_GET_LICENSE_INVALID_ARG PRL_RESULT_DECLARE_ERROR(137)
```

File

PrlErrorsValues.h

PRL_ERR_GET_MON_STATE_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_GET_MON_STATE_INVALID_ARG PRL_RESULT_DECLARE_ERROR(128)
```

File

PrlErrorsValues.h

PRL_ERR_GET_MON_STATE_VM_NOT_CONFIGURED Macro

Syntax

```
#define PRL_ERR_GET_MON_STATE_VM_NOT_CONFIGURED PRL_RESULT_DECLARE_ERROR(129)
```

File

PrlErrorsValues.h

PRL_ERR_GET_MON_STATE_VM_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_GET_MON_STATE_VM_NOT_CREATED PRL_RESULT_DECLARE_ERROR(127)
```

File

PrlErrorsValues.h

PRL_ERR_GET_NET_SERVICE_STATUS_FAILED Macro

Syntax

```
#define PRL_ERR_GET_NET_SERVICE_STATUS_FAILED PRL_RESULT_DECLARE_ERROR(353)
```

File

PrlErrorsValues.h

PRL_ERR_GET_RESOLUTION_TOOL_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_GET_RESOLUTION_TOOL_INVALID_ARG PRL_RESULT_DECLARE_ERROR(176)
```

File

PrlErrorsValues.h

PRL_ERR_GET_RESOLUTION_TOOL_VMNOTCREATED Macro

Syntax

```
#define PRL_ERR_GET_RESOLUTION_TOOL_VMNOTCREATED PRL_RESULT_DECLARE_ERROR(175)
```

File

PrlErrorsValues.h

PRL_ERR_GET_USER_HOME_DIR Macro

Syntax

```
#define PRL_ERR_GET_USER_HOME_DIR PRL_RESULT_DECLARE_ERROR(243)
```

File

PrlErrorsValues.h

PRL_ERR_GUEST_MAC_INVALID_VERSION Macro

Syntax

```
#define PRL_ERR_GUEST_MAC_INVALID_VERSION PRL_RESULT_DECLARE_ERROR(359)
```

File

PrlErrorsValues.h

PRL_ERR_GUEST_MAC_NOT_ENOUGH_MEMORY Macro

Syntax

```
#define PRL_ERR_GUEST_MAC_NOT_ENOUGH_MEMORY PRL_RESULT_DECLARE_ERROR(358)
```

File

PrlErrorsValues.h

PRL_ERR_GUEST_MAC_NOT_MACSERVER_HOST Macro

Syntax

```
#define PRL_ERR_GUEST_MAC_NOT_MACSERVER_HOST PRL_RESULT_DECLARE_ERROR(397)
```

File

PrlErrorsValues.h

PRL_ERR_GUEST_PROGRAM_EXECUTION_FAILED Macro

Syntax

```
#define PRL_ERR_GUEST_PROGRAM_EXECUTION_FAILED PRL_RESULT_DECLARE_ERROR(34004)
```

File

PrlErrorsValues.h

PRL_ERR_GUEST_TOOLS_NOT_INSTALLED Macro

Syntax

```
#define PRL_ERR_GUEST_TOOLS_NOT_INSTALLED PRL_RESULT_DECLARE_ERROR(423)
```

File

PrlErrorsValues.h

PRL_ERR_HANDSHAKE_FAILED Macro

Syntax

```
#define PRL_ERR_HANDSHAKE_FAILED PRL_RESULT_DECLARE_ERROR(412)
```

File

PrlErrorsValues.h

PRL_ERR_HARD_DISK_IMAGE_CORRUPTED Macro

Syntax

```
#define PRL_ERR_HARD_DISK_IMAGE_CORRUPTED PRL_RESULT_DECLARE_ERROR(2002)
```

File

PrlErrorsValues.h

PRL_ERR_HARD_DISK_NOT_VIRTUAL_OR_DISABLED Macro

Syntax

```
#define PRL_ERR_HARD_DISK_NOT_VIRTUAL_OR_DISABLED PRL_RESULT_DECLARE_ERROR(499)
```

File

PrlErrorsValues.h

PRL_ERR_HDD_IMAGE_CLONE_TO_SELF Macro

Syntax

```
#define PRL_ERR_HDD_IMAGE_CLONE_TO_SELF PRL_RESULT_DECLARE_ERROR(94)
```

File

PrlErrorsValues.h

PRL_ERR_HDD_IMAGE_COPY Macro

Syntax

```
#define PRL_ERR_HDD_IMAGE_COPY PRL_RESULT_DECLARE_ERROR(237)
```

File

PrlErrorsValues.h

PRL_ERR_HDD_IMAGE_IS_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_HDD_IMAGE_IS_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(288)
```

File

PrlErrorsValues.h

PRL_ERR_HDD_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_HDD_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(93)
```

File

PrlErrorsValues.h

PRL_ERR_HDD_IMAGE_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_HDD_IMAGE_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(206)
```

File

PrlErrorsValues.h

PRL_ERR_HOST_AMD_GUEST_MAC Macro

Syntax

```
#define PRL_ERR_HOST_AMD_GUEST_MAC PRL_RESULT_DECLARE_ERROR(348)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_AUTH_REQUIRED Macro

Syntax

```
#define PRL_ERR_HTTP_AUTH_REQUIRED PRL_RESULT_DECLARE_ERROR(38006)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_CONNECTION_REFUSED Macro

Syntax

```
#define PRL_ERR_HTTP_CONNECTION_REFUSED PRL_RESULT_DECLARE_ERROR(38001)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_HOST_NOT_FOUND Macro

Problem report send errors set +++ ALIGNED VALUE TO 38000

Syntax

```
#define PRL_ERR_HTTP_HOST_NOT_FOUND PRL_RESULT_DECLARE_ERROR(38000)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_INVALID_RESPONSE_HEADER Macro

Syntax

```
#define PRL_ERR_HTTP_INVALID_RESPONSE_HEADER PRL_RESULT_DECLARE_ERROR(38003)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_PROBLEM_REPORT_SEND_FAILURE Macro

Syntax

```
#define PRL_ERR_HTTP_PROBLEM_REPORT_SEND_FAILURE PRL_RESULT_DECLARE_ERROR(38007)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_PROXY_AUTH_REQUIRED Macro

Syntax

```
#define PRL_ERR_HTTP_PROXY_AUTH_REQUIRED PRL_RESULT_DECLARE_ERROR(38005)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_REQUEST_FAILED Macro

Syntax

```
#define PRL_ERR_HTTP_REQUEST_FAILED PRL_RESULT_DECLARE_ERROR(11715)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_UNEXPECTED_CLOSE Macro

Syntax

```
#define PRL_ERR_HTTP_UNEXPECTED_CLOSE PRL_RESULT_DECLARE_ERROR(38002)
```

File

PrlErrorsValues.h

PRL_ERR_HTTP_WRONG_CONTENT_LENGTH Macro

Syntax

```
#define PRL_ERR_HTTP_WRONG_CONTENT_LENGTH PRL_RESULT_DECLARE_ERROR(38004)
```

File

PrlErrorsValues.h

PRL_ERR_HVT_DISABLED Macro

Syntax

```
#define PRL_ERR_HVT_DISABLED PRL_RESULT_DECLARE_ERROR(341)
```

File

PrlErrorsValues.h

PRL_ERR_HVT_DISABLED_WARNING Macro

Syntax

```
#define PRL_ERR_HVT_DISABLED_WARNING PRL_RESULT_DECLARE_ERROR(507)
```

File

PrlErrorsValues.h

PRL_ERR_HVT_NOT_AVAILABLE_WARNING Macro

Syntax

```
#define PRL_ERR_HVT_NOT_AVAILABLE_WARNING PRL_RESULT_DECLARE_ERROR(483)
```

File

PrlErrorsValues.h

PRL_ERR_HVT_NOT_PRESENT Macro

Syntax

```
#define PRL_ERR_HVT_NOT_PRESENT PRL_RESULT_DECLARE_ERROR(336)
```

File

PrlErrorsValues.h

PRL_ERR_HVT_NOT_PRESENT_WARNING Macro

Syntax

```
#define PRL_ERR_HVT_NOT_PRESENT_WARNING PRL_RESULT_DECLARE_ERROR(36028)
```

File

PrlErrorsValues.h

PRL_ERR_HVT_TURNED_OFF_IN_CONFIG Macro

Syntax

```
#define PRL_ERR_HVT_TURNED_OFF_IN_CONFIG PRL_RESULT_DECLARE_ERROR(367)
```

File

PrlErrorsValues.h

PRL_ERR_HYP_ALLOC_VMS_MEMORY Macro

Syntax

```
#define PRL_ERR_HYP_ALLOC_VMS_MEMORY PRL_RESULT_DECLARE_ERROR(431)
```

File

PrlErrorsValues.h

PRL_ERR_IMAGE_NEED_TO_CONVERT Macro

Syntax

```
#define PRL_ERR_IMAGE_NEED_TO_CONVERT PRL_RESULT_DECLARE_ERROR(2001)
```

File

PrlErrorsValues.h

PRL_ERR_IMPERSONATE_FAILED Macro

Syntax

```
#define PRL_ERR_IMPERSONATE_FAILED PRL_RESULT_DECLARE_ERROR(350)
```

File

PrlErrorsValues.h

PRL_ERR_IMPORT_BOOTCAMP_VM_FAILED Macro

Syntax

```
#define PRL_ERR_IMPORT_BOOTCAMP_VM_FAILED PRL_RESULT_DECLARE_ERROR(41302)
```

File

PrlErrorsValues.h

PRL_ERR_IMPORT_BOOTCAMP_VM_NO_SPACE Macro

Syntax

```
#define PRL_ERR_IMPORT_BOOTCAMP_VM_NO_SPACE PRL_RESULT_DECLARE_ERROR(41303)
```

File

PrlErrorsValues.h

PRL_ERR_INCONSISTENCY_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_INCONSISTENCY_VM_CONFIG PRL_RESULT_DECLARE_ERROR(106)
```

File

PrlErrorsValues.h

PRL_ERR_INCORRECT_CDROM_PATH Macro

Syntax

```
#define PRL_ERR_INCORRECT_CDROM_PATH PRL_RESULT_DECLARE_ERROR(1001)
```

File

PrlErrorsValues.h

PRL_ERR_INCORRECT_FDD_PATH Macro

Syntax

```
#define PRL_ERR_INCORRECT_FDD_PATH PRL_RESULT_DECLARE_ERROR(335)
```

File

PrlErrorsValues.h

PRL_ERR_INCORRECT_PATH Macro

Syntax

```
#define PRL_ERR_INCORRECT_PATH PRL_RESULT_DECLARE_ERROR(324)
```

File

PrlErrorsValues.h

PRL_ERR_INSERTING_VM_TO_CATALOGUE Macro

Syntax

```
#define PRL_ERR_INSERTING_VM_TO_CATALOGUE PRL_RESULT_DECLARE_ERROR(238)
```

File

PrlErrorsValues.h

PRL_ERR_INSTALLATION_PROBLEM Macro

Syntax

```
#define PRL_ERR_INSTALLATION_PROBLEM PRL_RESULT_DECLARE_ERROR(284)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_ACCESS_TOKEN_RECEIVED Macro

Syntax

```
#define PRL_ERR_INVALID_ACCESS_TOKEN_RECEIVED PRL_RESULT_DECLARE_ERROR(245)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_ACTION_REQUESTED Macro

Syntax

```
#define PRL_ERR_INVALID_ACTION_REQUESTED PRL_RESULT_DECLARE_ERROR(516)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_INVALID_ARG PRL_RESULT_DECLARE_ERROR(3)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_CONFIGURATION Macro

Syntax

```
#define PRL_ERR_INVALID_CONFIGURATION PRL_RESULT_DECLARE_ERROR(365)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_CREATE_FLOPPY_IMAGE_PARAMETERS Macro

Syntax

```
#define PRL_ERR_INVALID_CREATE_FLOPPY_IMAGE_PARAMETERS PRL_RESULT_DECLARE_ERROR(551)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_HANDLE Macro

Syntax

```
#define PRL_ERR_INVALID_HANDLE PRL_RESULT_DECLARE_ERROR(4)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_HDD_GEOMETRY Macro

Syntax

```
#define PRL_ERR_INVALID_HDD_GEOMETRY PRL_RESULT_DECLARE_ERROR(207)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_KEXT_REBOOT_REQUIRED Macro

Syntax

```
#define PRL_ERR_INVALID_KEXT_REBOOT_REQUIRED PRL_RESULT_DECLARE_ERROR(430)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_MEMORY_GUARANTEE Macro

Syntax

```
#define PRL_ERR_INVALID_MEMORY_GUARANTEE PRL_RESULT_DECLARE_ERROR(537)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_MEMORY_QUOTA Macro

Syntax

```
#define PRL_ERR_INVALID_MEMORY_QUOTA PRL_RESULT_DECLARE_ERROR(536)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_MEMORY_SIZE Macro

Syntax

```
#define PRL_ERR_INVALID_MEMORY_SIZE PRL_RESULT_DECLARE_ERROR(204)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_OS_TYPE Macro

Syntax

```
#define PRL_ERR_INVALID_OS_TYPE PRL_RESULT_DECLARE_ERROR(10018)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_PARALLELS_DISK Macro

- Hard Drive

+++ ALIGNED VALUE TO 2000

Syntax

```
#define PRL_ERR_INVALID_PARALLELS_DISK PRL_RESULT_DECLARE_ERROR(2000)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_PARAM Macro

Syntax

```
#define PRL_ERR_INVALID_PARAM PRL_RESULT_DECLARE_ERROR(18)
```

File

PrlErrorsValues.h

PRL_ERR_INVALID_TOTAL_LIMIT Macro

Syntax

```
#define PRL_ERR_INVALID_TOTAL_LIMIT PRL_RESULT_DECLARE_ERROR(542)
```

File

PrlErrorsValues.h

PRL_ERR_IO_AUTHENTICATION_FAILED Macro

Syntax

```
#define PRL_ERR_IO_AUTHENTICATION_FAILED PRL_RESULT_DECLARE_ERROR(304)
```

File

PrlErrorsValues.h

PRL_ERR_IO_CONNECTION_TIMEOUT Macro

Syntax

```
#define PRL_ERR_IO_CONNECTION_TIMEOUT PRL_RESULT_DECLARE_ERROR(303)
```

File

PrlErrorsValues.h

PRL_ERR_IO_DISABLED Macro

Syntax

```
#define PRL_ERR_IO_DISABLED PRL_RESULT_DECLARE_ERROR(340)
```

File

PrlErrorsValues.h

PRL_ERR_IO_INVALID_POINTER_ACCESS Macro

Syntax

```
#define PRL_ERR_IO_INVALID_POINTER_ACCESS PRL_RESULT_DECLARE_ERROR(452)
```

File

PrlErrorsValues.h

PRL_ERR_IO_NO_CONNECTION Macro

Syntax

```
#define PRL_ERR_IO_NO_CONNECTION PRL_RESULT_DECLARE_ERROR(401)
```

File

PrlErrorsValues.h

PRL_ERR_IO_SEND_QUEUE_IS_FULL Macro

Syntax

```
#define PRL_ERR_IO_SEND_QUEUE_IS_FULL PRL_RESULT_DECLARE_ERROR(300)
```

File

PrlErrorsValues.h

PRL_ERR_IO_STOPPED Macro

Syntax

```
#define PRL_ERR_IO_STOPPED PRL_RESULT_DECLARE_ERROR(301)
```

File

PrlErrorsValues.h

PRL_ERR_IO_UNKNOWN_VM_ID Macro

Syntax

```
#define PRL_ERR_IO_UNKNOWN_VM_ID PRL_RESULT_DECLARE_ERROR(302)
```

File

PrlErrorsValues.h

PRL_ERR_IOSERVICE_COMPRESS Macro

Syntax

```
#define PRL_ERR_IOSERVICE_COMPRESS PRL_RESULT_DECLARE_ERROR(550)
```

File

PrlErrorsValues.h

PRL_ERR_IPC_ATTACH_FAILED Macro

Syntax

```
#define PRL_ERR_IPC_ATTACH_FAILED PRL_RESULT_DECLARE_ERROR(29010)
```

File

PrlErrorsValues.h

PRL_ERR_IPC_CREATE_MAP_FAILED Macro

Syntax

```
#define PRL_ERR_IPC_CREATE_MAP_FAILED PRL_RESULT_DECLARE_ERROR(29009)
```

File

PrlErrorsValues.h

PRL_ERR_IPC_FTOK_FAILED Macro

Syntax

```
#define PRL_ERR_IPC_FTOK_FAILED PRL_RESULT_DECLARE_ERROR(29011)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_INVALID_MODE Macro

Syntax

```
#define PRL_ERR_IPFO_INVALID_MODE PRL_RESULT_DECLARE_ERROR(29006)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_RECEIVE_FAILED Macro

Syntax

```
#define PRL_ERR_IPFO_RECEIVE_FAILED PRL_RESULT_DECLARE_ERROR(29008)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SEND_FAILED Macro

Syntax

```
#define PRL_ERR_IPFO_SEND_FAILED PRL_RESULT_DECLARE_ERROR(29007)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SOCKET_ACCEPT_FAILED Macro

Syntax

```
#define PRL_ERR_IPFO_SOCKET_ACCEPT_FAILED PRL_RESULT_DECLARE_ERROR(29003)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SOCKET_BIND_FAILED Macro

Syntax

```
#define PRL_ERR_IPFO_SOCKET_BIND_FAILED PRL_RESULT_DECLARE_ERROR(29001)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SOCKET_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_IPFO_SOCKET_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(29004)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SOCKET_CREATE_FAILED Macro

- Errors for IPC file open library

+++ ALIGNED VALUE TO 29000

Syntax

```
#define PRL_ERR_IPFO_SOCKET_CREATE_FAILED PRL_RESULT_DECLARE_ERROR(29000)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SOCKET_LISTEN_FAILED Macro

Syntax

```
#define PRL_ERR_IPFO_SOCKET_LISTEN_FAILED PRL_RESULT_DECLARE_ERROR(29002)
```

File

PrlErrorsValues.h

PRL_ERR_IPFO_SOCKET_NOT_OPENED Macro

Syntax

```
#define PRL_ERR_IPFO_SOCKET_NOT_OPENED PRL_RESULT_DECLARE_ERROR(29005)
```

File

PrlErrorsValues.h

PRL_ERR_IPHONE_PROXY_ALREADY_STARTED Macro

Syntax

```
#define PRL_ERR_IPHONE_PROXY_ALREADY_STARTED PRL_RESULT_DECLARE_ERROR(32002)
```

File

PrlErrorsValues.h

PRL_ERR_IPHONE_PROXY_CANNOT_START Macro

iPhone proxy errors +++ ALIGNED VALUE TO 32000

Syntax

```
#define PRL_ERR_IPHONE_PROXY_CANNOT_START PRL_RESULT_DECLARE_ERROR(32000)
```

File

PrlErrorsValues.h

PRL_ERR_IPHONE_PROXY_CANNOT_STOP Macro

Syntax

```
#define PRL_ERR_IPHONE_PROXY_CANNOT_STOP PRL_RESULT_DECLARE_ERROR(32001)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_ALREADY_REGISTERED Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_ALREADY_REGISTERED PRL_RESULT_DECLARE_ERROR(44413)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_CANNOT_CREATE_MOUNT_POINT Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_CANNOT_CREATE_MOUNT_POINT PRL_RESULT_DECLARE_ERROR(44414)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_CREATE Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_CREATE PRL_RESULT_DECLARE_ERROR(44402)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_EXTEND Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_EXTEND PRL_RESULT_DECLARE_ERROR(44406)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_GET_STATE Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_GET_STATE PRL_RESULT_DECLARE_ERROR(44407)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_INVALID_FSTYPE Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_INVALID_FSTYPE PRL_RESULT_DECLARE_ERROR(44411)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_LIBRARY Macro

iscsi storages errors +++ ALIGNED VALUE TO 44400

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_LIBRARY PRL_RESULT_DECLARE_ERROR(44400)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_MOUNT Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_MOUNT PRL_RESULT_DECLARE_ERROR(44404)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_MOUNT_POINT_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_MOUNT_POINT_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(44415)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_MOUNTED Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_MOUNTED PRL_RESULT_DECLARE_ERROR(44408)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(44412)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_NOT_MOUNTED Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_NOT_MOUNTED PRL_RESULT_DECLARE_ERROR(44409)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(44410)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_REMOVE Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_REMOVE PRL_RESULT_DECLARE_ERROR(44403)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_START Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_START PRL_RESULT_DECLARE_ERROR(44401)
```

File

PrlErrorsValues.h

PRL_ERR_ISCSI_STORAGE_UMOUNT Macro

Syntax

```
#define PRL_ERR_ISCSI_STORAGE_UMOUNT PRL_RESULT_DECLARE_ERROR(44405)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_COMPACT Macro

compact

Syntax

```
#define PRL_ERR_IT_CANT_COMPACT PRL_RESULT_DECLARE_ERROR(46219)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_COMPACT_LDM_DISK Macro

Syntax

```
#define PRL_ERR_IT_CANT_COMPACT_LDM_DISK PRL_RESULT_DECLARE_ERROR(46220)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_COMPACT_PLAIN_DISK Macro

Syntax

```
#define PRL_ERR_IT_CANT_COMPACT_PLAIN_DISK PRL_RESULT_DECLARE_ERROR(46222)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_GET_BITMAP Macro

Syntax

```
#define PRL_ERR_IT_CANT_GET_BITMAP PRL_RESULT_DECLARE_ERROR(46221)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_GET_PARAMS_FROM_OLD_IMAGE Macro

Syntax

```
#define PRL_ERR_IT_CANT_GET_PARAMS_FROM_OLD_IMAGE PRL_RESULT_DECLARE_ERROR(46204)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_OPEN_IMAGE Macro

Syntax

```
#define PRL_ERR_IT_CANT_OPEN_IMAGE PRL_RESULT_DECLARE_ERROR(46203)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_PREPARE_RECONFIG_DATA Macro

Syntax

```
#define PRL_ERR_IT_CANT_PREPARE_RECONFIG_DATA PRL_RESULT_DECLARE_ERROR(46209)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_RECONFIG Macro

Syntax

```
#define PRL_ERR_IT_CANT_RECONFIG PRL_RESULT_DECLARE_ERROR(46207)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_RECONFIG_BOOTCAMP Macro

Syntax

```
#define PRL_ERR_IT_CANT_RECONFIG_BOOTCAMP PRL_RESULT_DECLARE_ERROR(46208)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_RESIZE_LDM_DISK Macro

Syntax

```
#define PRL_ERR_IT_CANT_RESIZE_LDM_DISK PRL_RESULT_DECLARE_ERROR(46217)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_RESIZE_VOLUME Macro

resize

Syntax

```
#define PRL_ERR_IT_CANT_RESIZE_VOLUME PRL_RESULT_DECLARE_ERROR(46216)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CANT_VALIDATE_BOOTCAMP Macro

reconfig

Syntax

```
#define PRL_ERR_IT_CANT_VALIDATE_BOOTCAMP PRL_RESULT_DECLARE_ERROR(46206)
```

File

PrlErrorsValues.h

PRL_ERR_IT_CONVERT_TO_CURRENT_ERROR Macro

convert

Syntax

```
#define PRL_ERR_IT_CONVERT_TO_CURRENT_ERROR PRL_RESULT_DECLARE_ERROR(46214)
```

File

PrlErrorsValues.h

PRL_ERR_IT_DISK_SMALL_FOR_SPLIT Macro

Syntax

```
#define PRL_ERR_IT_DISK_SMALL_FOR_SPLIT PRL_RESULT_DECLARE_ERROR(46215)
```

File

PrlErrorsValues.h

PRL_ERR_IT_FAST_RESIZE_FAILURE Macro

Syntax

```
#define PRL_ERR_IT_FAST_RESIZE_FAILURE PRL_RESULT_DECLARE_ERROR(46101)
```

File

PrlErrorsValues.h

PRL_ERR_IT_FIRST_ERROR Macro

ALIGNED VALUE TO 46200

Syntax

```
#define PRL_ERR_IT_FIRST_ERROR PRL_RESULT_DECLARE_ERROR(46200)
```

File

PrlErrorsValues.h

PRL_ERR_IT_FIRST_FAILURE Macro

ALIGNED VALUE TO 46100

Syntax

```
#define PRL_ERR_IT_FIRST_FAILURE PRL_RESULT_DECLARE_ERROR(46100)
```

File

PrlErrorsValues.h

PRL_ERR_IT_FS_NOT_SUPPORTED_FOR_RESIZE Macro

Syntax

```
#define PRL_ERR_IT_FS_NOT_SUPPORTED_FOR_RESIZE PRL_RESULT_DECLARE_ERROR(46218)
```

File

PrlErrorsValues.h

PRL_ERR_IT_GUEST_TOOLS_UNKNOWN_STATE Macro

guest tools

Syntax

```
#define PRL_ERR_IT_GUEST_TOOLS_UNKNOWN_STATE PRL_RESULT_DECLARE_ERROR(46205)
```

File

PrlErrorsValues.h

PRL_ERR_IT_HAL_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_IT_HAL_NOT_FOUND PRL_RESULT_DECLARE_ERROR(46212)
```

File

PrlErrorsValues.h

PRL_ERR_IT_RECONFIG_PATH_NEEDED Macro

Syntax

```
#define PRL_ERR_IT_RECONFIG_PATH_NEEDED PRL_RESULT_DECLARE_ERROR(46210)
```

File

PrlErrorsValues.h

PRL_ERR_IT_ROLLBACK_FAILURE Macro

Syntax

```
#define PRL_ERR_IT_ROLLBACK_FAILURE PRL_RESULT_DECLARE_ERROR(46102)
```

File

PrlErrorsValues.h

PRL_ERR_IT_SIZE_CHANGED Macro

Syntax

```
#define PRL_ERR_IT_SIZE_CHANGED PRL_RESULT_DECLARE_ERROR(46202)
```

File

PrlErrorsValues.h

PRL_ERR_IT_SNAPSHOTS_MERGE_IS_NEEDED Macro

merge

Syntax

```
#define PRL_ERR_IT_SNAPSHOTS_MERGE_IS_NEEDED PRL_RESULT_DECLARE_ERROR(46213)
```

File

PrlErrorsValues.h

PRL_ERR_IT_UNKNOWN_OS Macro

Syntax

```
#define PRL_ERR_IT_UNKNOWN_OS PRL_RESULT_DECLARE_ERROR(46211)
```

File

PrlErrorsValues.h

PRL_ERR_IT_USER_INTERRUPTED Macro

Syntax

```
#define PRL_ERR_IT_USER_INTERRUPTED PRL_RESULT_DECLARE_ERROR(46201)
```

File

PrlErrorsValues.h

PRL_ERR_LIC_REGISTRATION_COMMON_ERROR Macro

Syntax

```
#define PRL_ERR_LIC_REGISTRATION_COMMON_ERROR PRL_RESULT_DECLARE_ERROR(11731)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_AUTH_FAILED Macro

Syntax

```
#define PRL_ERR_LICENSE_AUTH_FAILED PRL_RESULT_DECLARE_ERROR(11007)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_BETA_KEY_RELEASE_PRODUCT Macro

Syntax

```
#define PRL_ERR_LICENSE_BETA_KEY_RELEASE_PRODUCT PRL_RESULT_DECLARE_ERROR(11011)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_BLACKLISTED Macro

Syntax

```
#define PRL_ERR_LICENSE_BLACKLISTED PRL_RESULT_DECLARE_ERROR(11015)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_BLACKLISTED_TO_VM_OPERATION Macro

Syntax

```
#define PRL_ERR_LICENSE_BLACKLISTED_TO_VM_OPERATION PRL_RESULT_DECLARE_ERROR(11061)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_DEFERRED_LICENSE_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_LICENSE_DEFERRED_LICENSE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(11071)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_EXPIRED Macro

Syntax

```
#define PRL_ERR_LICENSE_EXPIRED PRL_RESULT_DECLARE_ERROR(11001)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_FILE_WRITE_FAILED Macro

Syntax

```
#define PRL_ERR_LICENSE_FILE_WRITE_FAILED PRL_RESULT_DECLARE_ERROR(11010)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_GRACED Macro

Syntax

```
#define PRL_ERR_LICENSE_GRACED PRL_RESULT_DECLARE_ERROR(11058)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_IS_NOT_VOLUME Macro

Syntax

```
#define PRL_ERR_LICENSE_IS_NOT_VOLUME PRL_RESULT_DECLARE_ERROR(11073)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_LICENSE_NOT_STARTED PRL_RESULT_DECLARE_ERROR(11014)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_NOT_VALID Macro

Syntax

```
#define PRL_ERR_LICENSE_NOT_VALID PRL_RESULT_DECLARE_ERROR(11000)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RELEASE_KEY_BETA_PRODUCT Macro

Syntax

```
#define PRL_ERR_LICENSE_RELEASE_KEY_BETA_PRODUCT PRL_RESULT_DECLARE_ERROR(11013)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_CPU_COUNT Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_CPU_COUNT PRL_RESULT_DECLARE_ERROR(11059)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_GUEST_OS Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_GUEST_OS PRL_RESULT_DECLARE_ERROR(11041)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_CLONE_VM Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_CLONE_VM PRL_RESULT_DECLARE_ERROR(11046)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_FROM_TEMPLATE Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_FROM_TEMPLATE PRL_RESULT_DECLARE_ERROR(11048)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_TO_TEMPLATE Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_TO_TEMPLATE PRL_RESULT_DECLARE_ERROR(11047)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_CREATE_VM Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_CREATE_VM PRL_RESULT_DECLARE_ERROR(11043)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_PAUSE_VM Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_PAUSE_VM PRL_RESULT_DECLARE_ERROR(11050)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_3RD_PARTY_VM Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_3RD_PARTY_VM PRL_RESULT_DECLARE_ERROR(11045)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_VM Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_VM PRL_RESULT_DECLARE_ERROR(11044)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT
Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT PRL_RESULT_DECLARE_ERROR(11042)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT_PER_USER
Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT_PER_USER
PRL_RESULT_DECLARE_ERROR(11060)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SAFEMODE_FEATURE
Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SAFEMODE_FEATURE PRL_RESULT_DECLARE_ERROR(11056)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SMARTGUARD_FEATURE

Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SMARTGUARD_FEATURE PRL_RESULT_DECLARE_ERROR(11057)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_CREATE

Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_CREATE PRL_RESULT_DECLARE_ERROR(11051)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_DELETE

Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_DELETE PRL_RESULT_DECLARE_ERROR(11053)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SHOW_TREE Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SHOW_TREE PRL_RESULT_DECLARE_ERROR(11054)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SWITCH Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SWITCH PRL_RESULT_DECLARE_ERROR(11052)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_SUSPEND_VM Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_SUSPEND_VM PRL_RESULT_DECLARE_ERROR(11049)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_RESTRICTED_TO_UNDODISK_FEATURE Macro

Syntax

```
#define PRL_ERR_LICENSE_RESTRICTED_TO_UNDODISK_FEATURE PRL_RESULT_DECLARE_ERROR(11055)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_TOO_MANY_MEMORY Macro

Syntax

```
#define PRL_ERR_LICENSE_TOO_MANY_MEMORY PRL_RESULT_DECLARE_ERROR(11016)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_TOO_MANY_VCPUS Macro

Syntax

```
#define PRL_ERR_LICENSE_TOO_MANY_VCPUS PRL_RESULT_DECLARE_ERROR(11012)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_UNSUPPORTED_LICENSE_TYPE_TO_DEACTIVATION Macro

Syntax

```
#define PRL_ERR_LICENSE_UNSUPPORTED_LICENSE_TYPE_TO_DEACTIVATION
PRL_RESULT_DECLARE_ERROR(11070)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_UPGRADE_NO_ACCEPTABLE_LICENSE Macro

Syntax

```
#define PRL_ERR_LICENSE_UPGRADE_NO_ACCEPTABLE_LICENSE
PRL_ERR_LICENSE_UPGRADE_NO_PERMANENT_LICENSE
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_UPGRADE_NO_PERMANENT_LICENSE Macro

Syntax

```
#define PRL_ERR_LICENSE_UPGRADE_NO_PERMANENT_LICENSE PRL_RESULT_DECLARE_ERROR(11009)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_VALID Macro

- License

+++ ALIGNED VALUE TO 11000

Syntax

```
#define PRL_ERR_LICENSE_VALID PRL_ERR_SUCCESS
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_VM_HAS_VTD_DEVICES Macro

Syntax

```
#define PRL_ERR_LICENSE_VM_HAS_VTD_DEVICES PRL_RESULT_DECLARE_ERROR(11017)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_WRONG_ADVANCED_FIELD Macro

Syntax

```
#define PRL_ERR_LICENSE_WRONG_ADVANCED_FIELD PRL_RESULT_DECLARE_ERROR(11008)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_WRONG_DISTRIBUTOR Macro

Syntax

```
#define PRL_ERR_LICENSE_WRONG_DISTRIBUTOR PRL_RESULT_DECLARE_ERROR(11006)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_WRONG_LANGUAGE Macro

Syntax

```
#define PRL_ERR_LICENSE_WRONG_LANGUAGE PRL_RESULT_DECLARE_ERROR(11005)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_WRONG_PLATFORM Macro

Syntax

```
#define PRL_ERR_LICENSE_WRONG_PLATFORM PRL_RESULT_DECLARE_ERROR(11004)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_WRONG_PRODUCT Macro

Syntax

```
#define PRL_ERR_LICENSE_WRONG_PRODUCT PRL_RESULT_DECLARE_ERROR(11003)
```

File

PrlErrorsValues.h

PRL_ERR_LICENSE_WRONG_VERSION Macro

Syntax

```
#define PRL_ERR_LICENSE_WRONG_VERSION PRL_RESULT_DECLARE_ERROR(11002)
```

File

PrlErrorsValues.h

PRL_ERR_LOCAL_AUTHENTICATION_FAILED Macro

Syntax

```
#define PRL_ERR_LOCAL_AUTHENTICATION_FAILED PRL_RESULT_DECLARE_ERROR(331)
```

File

PrlErrorsValues.h

PRL_ERR_LOGIN_BY_MISMATCH_CLIENT_MODE Macro

Syntax

```
#define PRL_ERR_LOGIN_BY_MISMATCH_CLIENT_MODE PRL_RESULT_DECLARE_ERROR(475)
```

File

PrlErrorsValues.h

PRL_ERR_LOW_MEMORY_LIMIT Macro

Syntax

```
#define PRL_ERR_LOW_MEMORY_LIMIT PRL_RESULT_DECLARE_ERROR(539)
```

File

PrlErrorsValues.h

PRL_ERR_LPT_DEVICE_ALREADY_EXIST Macro

Syntax

```
#define PRL_ERR_LPT_DEVICE_ALREADY_EXIST PRL_RESULT_DECLARE_ERROR(255)
```

File

PrlErrorsValues.h

PRL_ERR_LPT_OUTPUT_FILE_IS_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_LPT_OUTPUT_FILE_IS_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(216)
```

File

PrlErrorsValues.h

PRL_ERR_LPT_OUTPUT_FILE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_LPT_OUTPUT_FILE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(218)
```

File

PrlErrorsValues.h

PRL_ERR_MAC_ADDRESS_IN_CORRECT_LENGTH Macro

Syntax

```
#define PRL_ERR_MAC_ADDRESS_IN_CORRECT_LENGTH PRL_RESULT_DECLARE_ERROR(211)
```

File

PrlErrorsValues.h

PRL_ERR_MAC_ADDRESS_INCORRECT Macro

Syntax

```
#define PRL_ERR_MAC_ADDRESS_INCORRECT PRL_RESULT_DECLARE_ERROR(328)
```

File

PrlErrorsValues.h

PRL_ERR_MAC_ADDRESS_IS_EMPTY Macro

Syntax

```
#define PRL_ERR_MAC_ADDRESS_IS_EMPTY PRL_RESULT_DECLARE_ERROR(210)
```

File

PrlErrorsValues.h

PRL_ERR_MAC_ADDRESS_WITH_2_ZERO_START Macro

Syntax

```
#define PRL_ERR_MAC_ADDRESS_WITH_2_ZERO_START PRL_RESULT_DECLARE_ERROR(212)
```

File

PrlErrorsValues.h

PRL_ERR_MAC_ADDRESS_WITH_ALL_ZEROS Macro

Syntax

```
#define PRL_ERR_MAC_ADDRESS_WITH_ALL_ZEROS PRL_RESULT_DECLARE_ERROR(213)
```

File

PrlErrorsValues.h

PRL_ERR_MAC_ADDRESS_WITHIN_CORRECT_SYMBOLS Macro

Syntax

```
#define PRL_ERR_MAC_ADDRESS_WITHIN_CORRECT_SYMBOLS PRL_RESULT_DECLARE_ERROR(214)
```

File

PrlErrorsValues.h

PRL_ERR_MAKE_DIRECTORY Macro

Syntax

```
#define PRL_ERR_MAKE_DIRECTORY PRL_RESULT_DECLARE_ERROR(232)
```

File

PrlErrorsValues.h

PRL_ERR_MEM_EXCEED_PHY_SPACE Macro

Syntax

```
#define PRL_ERR_MEM_EXCEED_PHY_SPACE PRL_RESULT_DECLARE_ERROR(555)
```

File

PrlErrorsValues.h

PRL_ERR_MEMORY_ALLOC_ERROR Macro

Syntax

```
#define PRL_ERR_MEMORY_ALLOC_ERROR PRL_RESULT_DECLARE_ERROR(10013)
```

File

PrlErrorsValues.h

PRL_ERR_MERGE_XML_DOCUMENT_CONFLICT Macro

Syntax

```
#define PRL_ERR_MERGE_XML_DOCUMENT_CONFLICT PRL_RESULT_DECLARE_ERROR(515)
```

File

PrlErrorsValues.h

PRL_ERR_MOBILE_ADVANCED_AUTH_REQUIRED Macro

Syntax

```
#define PRL_ERR_MOBILE_ADVANCED_AUTH_REQUIRED PRL_RESULT_DECLARE_ERROR(558)
```

File

PrlErrorsValues.h

PRL_ERR_MOUNTER_LIST_NO_OBJECT Macro

Syntax

```
#define PRL_ERR_MOUNTER_LIST_NO_OBJECT PRL_RESULT_DECLARE_ERROR(46604)
```

File

PrlErrorsValues.h

PRL_ERR_MSG_START_32BIT_VM_ON_64BIT_HOST Macro

Syntax

```
#define PRL_ERR_MSG_START_32BIT_VM_ON_64BIT_HOST PRL_RESULT_DECLARE_ERROR(544)
```

File

PrlErrorsValues.h

PRL_ERR_NEED_KILL_PREV_SETTING Macro

Syntax

```
#define PRL_ERR_NEED_KILL_PREV_SETTING PRL_RESULT_DECLARE_SUCCESS(426)
```

File

PrlErrorsValues.h

PRL_ERR_NETWORK_ADAPTER_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_NETWORK_ADAPTER_NOT_FOUND PRL_RESULT_DECLARE_ERROR(485)
```

File

PrlErrorsValues.h

PRL_ERR_NETWORK_ROLLBACK_FAILED Macro

Syntax

```
#define PRL_ERR_NETWORK_ROLLBACK_FAILED PRL_RESULT_DECLARE_ERROR(270)
```

File

PrlErrorsValues.h

PRL_ERR_NO_BOOTING_DEVICE_SELECTED Macro

Syntax

```
#define PRL_ERR_NO_BOOTING_DEVICE_SELECTED PRL_RESULT_DECLARE_ERROR(334)
```

File

PrlErrorsValues.h

PRL_ERR_NO_CD_DRIVE_AVAILABLE Macro

Syntax

```
#define PRL_ERR_NO_CD_DRIVE_AVAILABLE PRL_RESULT_DECLARE_ERROR(418)
```

File

PrlErrorsValues.h

PRL_ERR_NO_DATA Macro

Syntax

```
#define PRL_ERR_NO_DATA PRL_RESULT_DECLARE_ERROR(14)
```

File

PrlErrorsValues.h

PRL_ERR_NO_DISP_CONFIG_FOUND Macro

Syntax

```
#define PRL_ERR_NO_DISP_CONFIG_FOUND PRL_RESULT_DECLARE_ERROR(19)
```

File

PrlErrorsValues.h

PRL_ERR_NO_GUEST_OS_FOUND Macro

Syntax

```
#define PRL_ERR_NO_GUEST_OS_FOUND PRL_RESULT_DECLARE_ERROR(45007)
```

File

PrlErrorsValues.h

PRL_ERR_NO_MORE_FREE_INTERFACE_SLOTS Macro

Syntax

```
#define PRL_ERR_NO_MORE_FREE_INTERFACE_SLOTS PRL_RESULT_DECLARE_ERROR(369)
```

File

PrlErrorsValues.h

PRL_ERR_NO_ONE_HARD_DISK_TO_COMPRESS Macro

Syntax

```
#define PRL_ERR_NO_ONE_HARD_DISK_TO_COMPRESS PRL_RESULT_DECLARE_SUCCESS(490)
```

File

PrlErrorsValues.h

PRL_ERR_NO_PROBLEM_REPORT_FOUND Macro

Syntax

```
#define PRL_ERR_NO_PROBLEM_REPORT_FOUND PRL_RESULT_DECLARE_ERROR(85)
```

File

PrlErrorsValues.h

PRL_ERR_NO_TARGET_DIR_PATH_SPECIFIED Macro

Syntax

```
#define PRL_ERR_NO_TARGET_DIR_PATH_SPECIFIED PRL_RESULT_DECLARE_ERROR(407)
```

File

PrlErrorsValues.h

PRL_ERR_NO_TARGET_PATH_SPECIFIED Macro

Syntax

```
#define PRL_ERR_NO_TARGET_PATH_SPECIFIED PRL_RESULT_DECLARE_ERROR(74)
```

File

PrlErrorsValues.h

PRL_ERR_NO_VM_DIR_CONFIG_FOUND Macro

Syntax

```
#define PRL_ERR_NO_VM_DIR_CONFIG_FOUND PRL_RESULT_DECLARE_ERROR(26)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_ALL_FILES_WAS_DELETED Macro

Syntax

```
#define PRL_ERR_NOT_ALL_FILES_WAS_DELETED PRL_RESULT_DECLARE_ERROR(102)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_CONNECTED_TO_DISPATCHER Macro

Syntax

```
#define PRL_ERR_NOT_CONNECTED_TO_DISPATCHER PRL_RESULT_DECLARE_ERROR(251)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_CONNECTED_TO_PROXY_MANAGER Macro

Syntax

```
#define PRL_ERR_NOT_CONNECTED_TO_PROXY_MANAGER PRL_RESULT_DECLARE_ERROR(564)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_ENOUGH_DISK_FREE_SPACE Macro

Syntax

```
#define PRL_ERR_NOT_ENOUGH_DISK_FREE_SPACE PRL_RESULT_DECLARE_ERROR(307)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_START_VM Macro

Syntax

```
#define PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_START_VM PRL_RESULT_DECLARE_ERROR(456)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_XML_SAVE Macro

Syntax

```
#define PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_XML_SAVE PRL_RESULT_DECLARE_ERROR(453)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_ENOUGH_PERMS_TO_OPEN_AUTHORIZATION_FILE Macro

Syntax

```
#define PRL_ERR_NOT_ENOUGH_PERMS_TO_OPEN_AUTHORIZATION_FILE PRL_RESULT_DECLARE_ERROR(298)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_ENOUGH_RIGHTS_FOR_LINKED_CLONE Macro

Syntax

```
#define PRL_ERR_NOT_ENOUGH_RIGHTS_FOR_LINKED_CLONE PRL_RESULT_DECLARE_ERROR(513)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_LOCK_OWNER_SESSION_TRIES_TO_UNLOCK Macro

Syntax

```
#define PRL_ERR_NOT_LOCK_OWNER_SESSION_TRIES_TO_UNLOCK PRL_RESULT_DECLARE_ERROR(478)
```

File

PrlErrorsValues.h

PRL_ERR_NOT_SENTILLION_CLIENT Macro

ALIGNED VALUE TO 39000

Syntax

```
#define PRL_ERR_NOT_SENTILLION_CLIENT PRL_RESULT_DECLARE_ERROR(39000)
```

File

PrlErrorsValues.h

PRL_ERR_NVRAM_FILE_COPY Macro

Syntax

```
#define PRL_ERR_NVRAM_FILE_COPY PRL_RESULT_DECLARE_ERROR(436)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_BAD_INTERFACE Macro

Syntax

```
#define PRL_ERR_OBJECT_BAD_INTERFACE PRL_RESULT_DECLARE_ERROR(42004)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_CLASS_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_OBJECT_CLASS_NOT_FOUND PRL_RESULT_DECLARE_ERROR(42008)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_DUPLICATE_CLASS Macro

Syntax

```
#define PRL_ERR_OBJECT_DUPLICATE_CLASS PRL_RESULT_DECLARE_ERROR(42007)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_DUPLICATE_UID Macro

Syntax

```
#define PRL_ERR_OBJECT_DUPLICATE_UID PRL_RESULT_DECLARE_ERROR(42003)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_LIB_CANT_GET_PERMS Macro

Syntax

```
#define PRL_ERR_OBJECT_LIB_CANT_GET_PERMS PRL_RESULT_DECLARE_ERROR(42006)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_LIB_LOAD_ERROR Macro

Syntax

```
#define PRL_ERR_OBJECT_LIB_LOAD_ERROR PRL_RESULT_DECLARE_ERROR(42001)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_LIB_NO_FUNCTIONS Macro

Syntax

```
#define PRL_ERR_OBJECT_LIB_NO_FUNCTIONS PRL_RESULT_DECLARE_ERROR(42002)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_LIB_WRONG_PERMS Macro

Syntax

```
#define PRL_ERR_OBJECT_LIB_WRONG_PERMS PRL_RESULT_DECLARE_ERROR(42005)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_NOT_FOUND Macro

- Shared objects manipulation errors

+++ ALIGNED VALUE TO 42000

Syntax

```
#define PRL_ERR_OBJECT_NOT_FOUND PRL_RESULT_DECLARE_ERROR(42000)
```

File

PrlErrorsValues.h

PRL_ERR_OBJECT_WAS_REMOVED Macro

Syntax

```
#define PRL_ERR_OBJECT_WAS_REMOVED PRL_RESULT_DECLARE_ERROR(406)
```

File

PrlErrorsValues.h

PRL_ERR_ONLY_ADMIN_CAN_SET_PARAMETER_STARTLOGINMODE_ROOT Macro

Syntax

```
#define PRL_ERR_ONLY_ADMIN_CAN_SET_PARAMETER_STARTLOGINMODE_ROOT PRL_RESULT_DECLARE_ERROR(395)
```

File

PrlErrorsValues.h

PRL_ERR_ONLY_ADMIN_CAN_SET_VERBOSE_LOGGING Macro

Syntax

```
#define PRL_ERR_ONLY_ADMIN_CAN_SET_VERBOSE_LOGGING PRL_RESULT_DECLARE_ERROR(527)
```

File

PrlErrorsValues.h

PRL_ERR_ONLY_ADMIN_OR_VM_OWNER_CAN_OPEN_THIS_SESSION Macro

VM guest operations errors +++ ALIGNED VALUE TO 34000

Syntax

```
#define PRL_ERR_ONLY_ADMIN_OR_VM_OWNER_CAN_OPEN_THIS_SESSION PRL_RESULT_DECLARE_ERROR(34000)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_DISP_CONFIG_READ Macro

Syntax

```
#define PRL_ERR_OPEN_DISP_CONFIG_READ PRL_RESULT_DECLARE_ERROR(21)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_DISP_CONFIG_WRITE Macro

Syntax

```
#define PRL_ERR_OPEN_DISP_CONFIG_WRITE PRL_RESULT_DECLARE_ERROR(22)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_FAILED Macro

Syntax

```
#define PRL_ERR_OPEN_FAILED PRL_RESULT_DECLARE_ERROR(520)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_PROBLEM_REPORT_READ Macro

Syntax

```
#define PRL_ERR_OPEN_PROBLEM_REPORT_READ PRL_RESULT_DECLARE_ERROR(86)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_VM_CONFIG_READ Macro

Syntax

```
#define PRL_ERR_OPEN_VM_CONFIG_READ PRL_RESULT_DECLARE_ERROR(33)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_VM_CONFIG_WRITE Macro

Syntax

```
#define PRL_ERR_OPEN_VM_CONFIG_WRITE PRL_RESULT_DECLARE_ERROR(34)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_VM_DIR_CONFIG_READ Macro

Syntax

```
#define PRL_ERR_OPEN_VM_DIR_CONFIG_READ PRL_RESULT_DECLARE_ERROR(28)
```

File

PrlErrorsValues.h

PRL_ERR_OPEN_VM_DIR_CONFIG_WRITE Macro

Syntax

```
#define PRL_ERR_OPEN_VM_DIR_CONFIG_WRITE PRL_RESULT_DECLARE_ERROR(29)
```

File

PrlErrorsValues.h

PRL_ERR_OPERATION_FAILED Macro

Syntax

```
#define PRL_ERR_OPERATION_FAILED PRL_RESULT_DECLARE_ERROR(16)
```

File

PrlErrorsValues.h

PRL_ERR_OPERATION_PENDING Macro

Syntax

```
#define PRL_ERR_OPERATION_PENDING PRL_RESULT_DECLARE_ERROR(13)
```

File

PrlErrorsValues.h

PRL_ERR_OPERATION_WAS_CANCELED Macro

Syntax

```
#define PRL_ERR_OPERATION_WAS_CANCELED PRL_RESULT_DECLARE_ERROR(275)
```

File

PrlErrorsValues.h

PRL_ERR_OS_RECONFIG_DATA_ABSENT Macro

Syntax

```
#define PRL_ERR_OS_RECONFIG_DATA_ABSENT PRL_RESULT_DECLARE_ERROR(45010)
```

File

PrlErrorsValues.h

PRL_ERR_OUT_OF_DISK_SPACE Macro

Syntax

```
#define PRL_ERR_OUT_OF_DISK_SPACE PRL_RESULT_DECLARE_ERROR(297)
```

File

PrlErrorsValues.h

PRL_ERR_OUT_OF_MEMORY Macro

Syntax

```
#define PRL_ERR_OUT_OF_MEMORY PRL_RESULT_DECLARE_ERROR(2)
```

File

PrlErrorsValues.h

PRL_ERR_PARALLEL_PORT_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_PARALLEL_PORT_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(97)
```

File

PrlErrorsValues.h

PRL_ERR_PARALLEL_PORT_IMG_COPY Macro

Syntax

```
#define PRL_ERR_PARALLEL_PORT_IMG_COPY PRL_RESULT_DECLARE_ERROR(236)
```

File

PrlErrorsValues.h

PRL_ERR_PARALLEL_PORT_NO_RASTERIZER Macro

Syntax

```
#define PRL_ERR_PARALLEL_PORT_NO_RASTERIZER PRL_RESULT_DECLARE_ERROR(512)
```

File

PrlErrorsValues.h

PRL_ERR_PARAM_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_PARAM_NOT_FOUND PRL_RESULT_DECLARE_ERROR(17)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_CLIENT_PREFS Macro

Syntax

```
#define PRL_ERR_PARSE_CLIENT_PREFS PRL_RESULT_DECLARE_ERROR(117)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_COMMON_SERVER_PREFS Macro

Syntax

```
#define PRL_ERR_PARSE_COMMON_SERVER_PREFS PRL_RESULT_DECLARE_ERROR(121)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_DISP_CONFIG Macro

Syntax

```
#define PRL_ERR_PARSE_DISP_CONFIG PRL_RESULT_DECLARE_ERROR(23)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_FILESYSTEM_INFO Macro

Syntax

```
#define PRL_ERR_PARSE_FILESYSTEM_INFO PRL_RESULT_DECLARE_ERROR(72)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_HARD_DISK_HW_INFO Macro

Syntax

```
#define PRL_ERR_PARSE_HARD_DISK_HW_INFO PRL_RESULT_DECLARE_ERROR(71)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_HOST_HW_INFO Macro

Syntax

```
#define PRL_ERR_PARSE_HOST_HW_INFO PRL_RESULT_DECLARE_ERROR(70)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_PROBLEM_REPORT Macro

Syntax

```
#define PRL_ERR_PARSE_PROBLEM_REPORT PRL_RESULT_DECLARE_ERROR(84)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_STATISTICS Macro

Syntax

```
#define PRL_ERR_PARSE_STATISTICS PRL_RESULT_DECLARE_ERROR(276)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_USER_PROFILE Macro

Syntax

```
#define PRL_ERR_PARSE_USER_PROFILE PRL_RESULT_DECLARE_ERROR(73)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_PARSE_VM_CONFIG PRL_RESULT_DECLARE_ERROR(36)
```

File

PrlErrorsValues.h

PRL_ERR_PARSE_VM_DIR_CONFIG Macro

Syntax

```
#define PRL_ERR_PARSE_VM_DIR_CONFIG PRL_RESULT_DECLARE_ERROR(30)
```

File

PrlErrorsValues.h

PRL_ERR_PARSING_EVENT Macro

Syntax

```
#define PRL_ERR_PARSING_EVENT PRL_RESULT_DECLARE_ERROR(24)
```

File

PrlErrorsValues.h

PRL_ERR_PATH_IS_NOT_DIRECTORY Macro

Syntax

```
#define PRL_ERR_PATH_IS_NOT_DIRECTORY PRL_RESULT_DECLARE_ERROR(37031)
```

File

PrlErrorsValues.h

PRL_ERR_PAX_ACCESS_FORBIDDEN Macro

Syntax

```
#define PRL_ERR_PAX_ACCESS_FORBIDDEN PRL_RESULT_DECLARE_ERROR(612)
```

File

PrlErrorsValues.h

PRL_ERR_PAX_HOST_LIMIT_WAS_EXCEEDED Macro

Syntax

```
#define PRL_ERR_PAX_HOST_LIMIT_WAS_EXCEEDED PRL_RESULT_DECLARE_ERROR(604)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_EXEC_FAILED Macro

Syntax

```
#define PRL_ERR_PCMOVER_EXEC_FAILED PRL_RESULT_DECLARE_ERROR(43103)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_MIGRATE_FAILED Macro

Syntax

```
#define PRL_ERR_PCMOVER_MIGRATE_FAILED PRL_RESULT_DECLARE_ERROR(43102)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_NOT_INSTALLED Macro

PCmover managment library errors +++ ALIGNED VALUE TO 43100

Syntax

```
#define PRL_ERR_PCMOVER_NOT_INSTALLED PRL_RESULT_DECLARE_ERROR(43100)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_POLICY_FILE_OPEN_ERROR Macro

Syntax

```
#define PRL_ERR_PCMOVER_POLICY_FILE_OPEN_ERROR PRL_RESULT_DECLARE_ERROR(43105)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_POLICY_FILE_WRITE_ERROR Macro

Syntax

```
#define PRL_ERR_PCMOVER_POLICY_FILE_WRITE_ERROR PRL_RESULT_DECLARE_ERROR(43104)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_VAN_FILE_CREATE_ERROR Macro

Syntax

```
#define PRL_ERR_PCMOVER_VAN_FILE_CREATE_ERROR PRL_RESULT_DECLARE_ERROR(43106)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_VAN_FILE_OPEN_ERROR Macro

Syntax

```
#define PRL_ERR_PCMOVER_VAN_FILE_OPEN_ERROR PRL_RESULT_DECLARE_ERROR(43107)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_VAN_FILE_PREPARE_FAILED Macro

Syntax

```
#define PRL_ERR_PCMOVER_VAN_FILE_PREPARE_FAILED PRL_RESULT_DECLARE_ERROR(43101)
```

File

PrlErrorsValues.h

PRL_ERR_PCMOVER_WINDOWS_DIR_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_PCMOVER_WINDOWS_DIR_NOT_EXIST PRL_RESULT_DECLARE_ERROR(43108)
```

File

PrlErrorsValues.h

PRL_ERR_PERFORM_MOUNTER_COMMAND Macro

Syntax

```
#define PRL_ERR_PERFORM_MOUNTER_COMMAND PRL_RESULT_DECLARE_ERROR(46603)
```

File

PrlErrorsValues.h

PRL_ERR_PLAYER_CANT_SUSPEND_IN_PAUSE Macro

Syntax

```
#define PRL_ERR_PLAYER_CANT_SUSPEND_IN_PAUSE PRL_RESULT_DECLARE_ERROR(481)
```

File

PrlErrorsValues.h

PRL_ERR_PPC_INVALID_CREDENTIALS Macro

- Proxy connection errors

+++ ALIGNED VALUE TO 600

Syntax

```
#define PRL_ERR_PPC_INVALID_CREDENTIALS PRL_RESULT_DECLARE_ERROR(600)
```

File

PrlErrorsValues.h

PRL_ERR_PPC_INVALID_CREDENTIALS_ON_RECONNECT Macro

Syntax

```
#define PRL_ERR_PPC_INVALID_CREDENTIALS_ON_RECONNECT PRL_RESULT_DECLARE_ERROR(601)
```

File

PrlErrorsValues.h

PRL_ERR_PPC_SERVER_BUSY Macro

Syntax

```
#define PRL_ERR_PPC_SERVER_BUSY PRL_RESULT_DECLARE_ERROR(602)
```

File

PrlErrorsValues.h

PRL_ERR_PPC_UNABLE_TO_CONNECT Macro

Syntax

```
#define PRL_ERR_PPC_UNABLE_TO_CONNECT PRL_RESULT_DECLARE_ERROR(603)
```

File

PrlErrorsValues.h

PRL_ERR_PREPARE_FOR_HIBERNATE_TASK_ALREADY_RUN Macro

Syntax

```
#define PRL_ERR_PREPARE_FOR_HIBERNATE_TASK_ALREADY_RUN PRL_RESULT_DECLARE_ERROR(464)
```

File

PrlErrorsValues.h

PRL_ERR_PREPARE_FOR_HIBERNATE_VM_CANNOT_STAND_BY Macro

Syntax

```
#define PRL_ERR_PREPARE_FOR_HIBERNATE_VM_CANNOT_STAND_BY PRL_RESULT_DECLARE_ERROR(467)
```

File

PrlErrorsValues.h

PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WITHOUT_TOOLS Macro

Syntax

```
#define PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WITHOUT_TOOLS PRL_RESULT_DECLARE_ERROR(466)
```

File

PrlErrorsValues.h

PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WRONG_STATE Macro

Syntax

```
#define PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WRONG_STATE PRL_RESULT_DECLARE_ERROR(465)
```

File

PrlErrorsValues.h

PRL_ERR_PREV_SESSION_IS_ACTIVE Macro

Syntax

```
#define PRL_ERR_PREV_SESSION_IS_ACTIVE PRL_RESULT_DECLARE_ERROR(64)
```

File

PrlErrorsValues.h

PRL_ERR_PROBLEM_REPORT_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_PROBLEM_REPORT_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(88)
```

File

PrlErrorsValues.h

PRL_ERR_PROBLEM_REPORT_FILE_NOTSET Macro

Syntax

```
#define PRL_ERR_PROBLEM_REPORT_FILE_NOTSET PRL_RESULT_DECLARE_ERROR(87)
```

File

PrlErrorsValues.h

PRL_ERR_PROBLEM_REPORT_WRITE Macro

Syntax

```
#define PRL_ERR_PROBLEM_REPORT_WRITE PRL_RESULT_DECLARE_ERROR(89)
```

File

PrlErrorsValues.h

PRL_ERR_PROXY_HANDSHAKE_FAILED Macro

Syntax

```
#define PRL_ERR_PROXY_HANDSHAKE_FAILED PRL_RESULT_DECLARE_ERROR(522)
```

File

PrlErrorsValues.h

PRL_ERR_PROXY_PEER_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_PROXY_PEER_NOT_FOUND PRL_RESULT_DECLARE_ERROR(525)
```

File

PrlErrorsValues.h

PRL_ERR_PROXY_WRONG_PORT_NUMBER Macro

Syntax

```
#define PRL_ERR_PROXY_WRONG_PORT_NUMBER PRL_RESULT_DECLARE_ERROR(524)
```

File

PrlErrorsValues.h

PRL_ERR_PROXY_WRONG_PROTOCOL_VERSION Macro

Syntax

```
#define PRL_ERR_PROXY_WRONG_PROTOCOL_VERSION PRL_RESULT_DECLARE_ERROR(523)
```

File

PrlErrorsValues.h

PRL_ERR_READ_FAILED Macro

Syntax

```
#define PRL_ERR_READ_FAILED PRL_RESULT_DECLARE_ERROR(518)
```

File

PrlErrorsValues.h

PRL_ERR_READ_XML_CONTENT Macro

Syntax

```
#define PRL_ERR_READ_XML_CONTENT PRL_RESULT_DECLARE_ERROR(35)
```

File

PrlErrorsValues.h

PRL_ERR_READONLY_FILESYSTEM Macro

Syntax

```
#define PRL_ERR_READONLY_FILESYSTEM PRL_RESULT_DECLARE_ERROR(356)
```

File

PrlErrorsValues.h

PRL_ERR_REBOOT_HOST Macro

Syntax

```
#define PRL_ERR_REBOOT_HOST PRL_RESULT_DECLARE_ERROR(352)
```

File

PrlErrorsValues.h

PRL_ERR_REG_PSTORAGE_REVOKE_IS_NEEDS Macro

Syntax

```
#define PRL_ERR_REG_PSTORAGE_REVOKE_IS_NEEDS PRL_RESULT_DECLARE_ERROR(46901)
```

File

PrlErrorsValues.h

PRL_ERR_REMOTE_DEVICE_EXIST Macro

Syntax

```
#define PRL_ERR_REMOTE_DEVICE_EXIST PRL_RESULT_DECLARE_ERROR(279)
```

File

PrlErrorsValues.h

PRL_ERR_REMOTE_DEVICE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_REMOTE_DEVICE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(280)
```

File

PrlErrorsValues.h

PRL_ERR_REMOTE_DEVICE_NOT_REGISTERED Macro

Device of this type cannot be remote device type is not registered

Syntax

```
#define PRL_ERR_REMOTE_DEVICE_NOT_REGISTERED PRL_RESULT_DECLARE_ERROR(278)
```

File

PrlErrorsValues.h

PRL_ERR_REMOTE_DISPLAY_EMPTY_PASSWORD Macro

Syntax

```
#define PRL_ERR_REMOTE_DISPLAY_EMPTY_PASSWORD PRL_RESULT_DECLARE_ERROR(487)
```

File

PrlErrorsValues.h

PRL_ERR_REMOTE_DISPLAY_HOST_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_REMOTE_DISPLAY_HOST_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(344)
```

File

PrlErrorsValues.h

PRL_ERR_REMOTE_DISPLAY_WRONG_PORT_NUMBER Macro

Syntax

```
#define PRL_ERR_REMOTE_DISPLAY_WRONG_PORT_NUMBER PRL_RESULT_DECLARE_ERROR(345)
```

File

PrlErrorsValues.h

PRL_ERR_RESUME_BOOTCAMP_CHANGED_DISK_CONTENTS Macro

Syntax

```
#define PRL_ERR_RESUME_BOOTCAMP_CHANGED_DISK_CONTENTS PRL_RESULT_DECLARE_ERROR(20014)
```

File

PrlErrorsValues.h

PRL_ERR_RESUME_BOOTCAMP_CORRUPT_DISK_STATE_PARAM Macro

Syntax

```
#define PRL_ERR_RESUME_BOOTCAMP_CORRUPT_DISK_STATE_PARAM PRL_RESULT_DECLARE_ERROR(20013)
```

File

PrlErrorsValues.h

PRL_ERR_RETRIEVE_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_RETRIEVE_VM_CONFIG PRL_RESULT_DECLARE_ERROR(240)
```

File

PrlErrorsValues.h

PRL_ERR_RETURN_CODE_RANG_EEND Macro

Syntax

```
#define PRL_ERR_RETURN_CODE_RANG_EEND PRL_RESULT_DECLARE_ERROR(113)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_IMPERSONATE_FAILED Macro

Syntax

```
#define PRL_ERR_REVERT_IMPERSONATE_FAILED PRL_RESULT_DECLARE_ERROR(351)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT Macro

Syntax

```
#define PRL_ERR_REVERT_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT PRL_RESULT_DECLARE_ERROR(36011)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_SNAPSHOT_VM_VTD_PAUSED Macro

Syntax

```
#define PRL_ERR_REVERT_SNAPSHOT_VM_VTD_PAUSED PRL_RESULT_DECLARE_ERROR(36010)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL
PRL_RESULT_DECLARE_ERROR(36014)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL
PRL_RESULT_DECLARE_ERROR(36013)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS Macro

Syntax

```
#define PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS PRL_RESULT_DECLARE_ERROR(36027)
```

File

PrlErrorsValues.h

PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL
PRL_RESULT_DECLARE_ERROR(36012)
```

File

PrlErrorsValues.h

PRL_ERR_RUN_VM_ACTION_SCRIPT Macro

Syntax

```
#define PRL_ERR_RUN_VM_ACTION_SCRIPT PRL_RESULT_DECLARE_ERROR(552)
```

File

PrlErrorsValues.h

PRL_ERR_RUNNING_VM_OR_CT Macro

Syntax

```
#define PRL_ERR_RUNNING_VM_OR_CT PRL_RESULT_DECLARE_ERROR(54103)
```

File

PrlErrorsValues.h

PRL_ERR_SAFE_MODE_START_DURING_SUSPENDING_SYNC Macro

Syntax

```
#define PRL_ERR_SAFE_MODE_START_DURING_SUSPENDING_SYNC PRL_RESULT_DECLARE_ERROR(20011)
```

File

PrlErrorsValues.h

PRL_ERR_SAMPLE_CONFIG_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_SAMPLE_CONFIG_NOT_FOUND PRL_RESULT_DECLARE_ERROR(27029)
```

File

PrlErrorsValues.h

PRL_ERR_SAVE_VM_CATALOG Macro

Syntax

```
#define PRL_ERR_SAVE_VM_CATALOG PRL_RESULT_DECLARE_ERROR(90)
```

File

PrlErrorsValues.h

PRL_ERR_SAVE_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_SAVE_VM_CONFIG PRL_RESULT_DECLARE_ERROR(234)
```

File

PrlErrorsValues.h

PRL_ERR_SDK_TRY_AGAIN Macro

SDK errors +++ ALIGNED VALUE TO 50000

Syntax

```
#define PRL_ERR_SDK_TRY_AGAIN PRL_RESULT_DECLARE_ERROR(50000)
```

File

PrlErrorsValues.h

PRL_ERR_SEARCH_CONFIG_OPERATION_CANCELED Macro

Syntax

```
#define PRL_ERR_SEARCH_CONFIG_OPERATION_CANCELED PRL_RESULT_DECLARE_ERROR(108)
```

File

PrlErrorsValues.h

PRL_ERR_SEND_COMMAND_TOWS_FAILED Macro

Syntax

```
#define PRL_ERR_SEND_COMMAND_TOWS_FAILED PRL_RESULT_DECLARE_ERROR(250)
```

File

PrlErrorsValues.h

PRL_ERR_SERIAL_IMG_COPY Macro

Syntax

```
#define PRL_ERR_SERIAL_IMG_COPY PRL_RESULT_DECLARE_ERROR(235)
```

File

PrlErrorsValues.h

PRL_ERR_SERIAL_IMG_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_SERIAL_IMG_NOT_FOUND PRL_RESULT_DECLARE_ERROR(233)
```

File

PrlErrorsValues.h

PRL_ERR_SERIAL_PORT_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_SERIAL_PORT_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(96)
```

File

PrlErrorsValues.h

PRL_ERR_SERVER_GOES_DOWN Macro

Syntax

```
#define PRL_ERR_SERVER_GOES_DOWN PRL_RESULT_DECLARE_ERROR(248)
```

File

PrlErrorsValues.h

PRL_ERR_SERVER_PREFS_EDIT_COLLISION Macro

Syntax

```
#define PRL_ERR_SERVER_PREFS_EDIT_COLLISION PRL_RESULT_DECLARE_ERROR(326)
```

File

PrlErrorsValues.h

PRL_ERR_SERVICE_BUSY Macro

Syntax

```
#define PRL_ERR_SERVICE_BUSY PRL_RESULT_DECLARE_ERROR(521)
```

File

PrlErrorsValues.h

PRL_ERR_SET_CPULIMIT Macro

Virtuozzo specific error +++ ALIGNED VALUE TO 44000

Syntax

```
#define PRL_ERR_SET_CPULIMIT PRL_RESULT_DECLARE_ERROR(44000)
```

File

PrlErrorsValues.h

PRL_ERR_SET_CPUMASK Macro

Syntax

```
#define PRL_ERR_SET_CPUMASK PRL_RESULT_DECLARE_ERROR(44005)
```

File

PrlErrorsValues.h

PRL_ERR_SET_CPUUNITS Macro

Syntax

```
#define PRL_ERR_SET_CPUUNITS PRL_RESULT_DECLARE_ERROR(44001)
```

File

PrlErrorsValues.h

PRL_ERR_SET_IOLIMIT Macro

Syntax

```
#define PRL_ERR_SET_IOLIMIT PRL_RESULT_DECLARE_ERROR(44004)
```

File

PrlErrorsValues.h

PRL_ERR_SET_IOPRIO Macro

Syntax

```
#define PRL_ERR_SET_IOPRIO PRL_RESULT_DECLARE_ERROR(44002)
```

File

PrlErrorsValues.h

PRL_ERR_SET_LICENSE_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_SET_LICENSE_INVALID_ARG PRL_RESULT_DECLARE_ERROR(138)
```

File

PrlErrorsValues.h

PRL_ERR_SET_LICENSE_INVALID_KEY Macro

Syntax

```
#define PRL_ERR_SET_LICENSE_INVALID_KEY PRL_RESULT_DECLARE_ERROR(139)
```

File

PrlErrorsValues.h

PRL_ERR_SET_NETWORK_SETTINGS_FAILED Macro

Syntax

```
#define PRL_ERR_SET_NETWORK_SETTINGS_FAILED PRL_RESULT_DECLARE_ERROR(34005)
```

File

PrlErrorsValues.h

PRL_ERR_SHUT_DOWN_NOT_IFICATION_SERVICE Macro

Syntax

```
#define PRL_ERR_SHUT_DOWN_NOT_IFICATION_SERVICE PRL_RESULT_DECLARE_ERROR(247)
```

File

PrlErrorsValues.h

PRL_ERR_SMC_ERROR Macro

Syntax

```
#define PRL_ERR_SMC_ERROR PRL_RESULT_DECLARE_ERROR(111)
```

File

PrlErrorsValues.h

PRL_ERR_SMP_NOT_SUPPORTED_HVT_DISABLED Macro

Syntax

```
#define PRL_ERR_SMP_NOT_SUPPORTED_HVT_DISABLED PRL_RESULT_DECLARE_ERROR(505)
```

File

PrlErrorsValues.h

PRL_ERR_SMP_NOT_SUPPORTED_HVT_NOT_PRESENT Macro

Syntax

```
#define PRL_ERR_SMP_NOT_SUPPORTED_HVT_NOT_PRESENT PRL_RESULT_DECLARE_ERROR(506)
```

File

PrlErrorsValues.h

PRL_ERR_SNAPSHOTS_COPY Macro

Syntax

```
#define PRL_ERR_SNAPSHOTS_COPY PRL_RESULT_DECLARE_ERROR(437)
```

File

PrlErrorsValues.h

PRL_ERR_SOME_TASKS_PRESENT Macro

Syntax

```
#define PRL_ERR_SOME_TASKS_PRESENT PRL_RESULT_DECLARE_ERROR(363)
```

File

PrlErrorsValues.h

PRL_ERR_SOME_VMS_RUNNING Macro

Syntax

```
#define PRL_ERR_SOME_VMS_RUNNING PRL_RESULT_DECLARE_ERROR(362)
```

File

PrlErrorsValues.h

PRL_ERR_SOUND_BAD_EMULATION_TYPE Macro

Syntax

```
#define PRL_ERR_SOUND_BAD_EMULATION_TYPE PRL_RESULT_DECLARE_ERROR(5003)
```

File

PrlErrorsValues.h

PRL_ERR_SOUND_DEVICE_WRITE_FAILED Macro

- Sound

+++ ALIGNED VALUE TO 5000

Syntax

```
#define PRL_ERR_SOUND_DEVICE_WRITE_FAILED PRL_RESULT_DECLARE_ERROR(5000)
```

File

PrlErrorsValues.h

PRL_ERR_SOUND_IN_DEVICE_OPEN_FAILED Macro

Syntax

```
#define PRL_ERR_SOUND_IN_DEVICE_OPEN_FAILED PRL_RESULT_DECLARE_ERROR(5002)
```

File

PrlErrorsValues.h

PRL_ERR_SOUND_OUT_DEVICE_OPEN_FAILED Macro

Syntax

```
#define PRL_ERR_SOUND_OUT_DEVICE_OPEN_FAILED PRL_RESULT_DECLARE_ERROR(5001)
```

File

PrlErrorsValues.h

PRL_ERR_SOURCE_ISNT_BOOTCAMP_DISK Macro

Syntax

```
#define PRL_ERR_SOURCE_ISNT_BOOTCAMP_DISK PRL_RESULT_DECLARE_ERROR(45009)
```

File

PrlErrorsValues.h

PRL_ERR_SSL_HANDSHAKE_FAILED Macro

Syntax

```
#define PRL_ERR_SSL_HANDSHAKE_FAILED PRL_RESULT_DECLARE_ERROR(414)
```

File

PrlErrorsValues.h

PRL_ERR_STAND_BY_VM_BY_GUEST_SLEEP_TIMEOUT Macro

Syntax

```
#define PRL_ERR_STAND_BY_VM_BY_GUEST_SLEEP_TIMEOUT PRL_RESULT_DECLARE_ERROR(36019)
```

File

PrlErrorsValues.h

PRL_ERR_STAND_BY_VM_PAUSED Macro

Syntax

```
#define PRL_ERR_STAND_BY_VM_PAUSED PRL_RESULT_DECLARE_ERROR(36015)
```

File

PrlErrorsValues.h

PRL_ERR_STAND_BY_VM_WITH_OUTDATED_SHUTDOWN_TOOL
Macro

Syntax

```
#define PRL_ERR_STAND_BY_VM_WITH_OUTDATED_SHUTDOWN_TOOL PRL_RESULT_DECLARE_ERROR(36017)
```

File

PrlErrorsValues.h

PRL_ERR_STAND_BY_VM_WITH_UNLOADED_SHUTDOWN_TOOL
Macro

Syntax

```
#define PRL_ERR_STAND_BY_VM_WITH_UNLOADED_SHUTDOWN_TOOL PRL_RESULT_DECLARE_ERROR(36016)
```

File

PrlErrorsValues.h

PRL_ERR_STAND_BY_VM_WITH_UNSUPPORTED_SHUTDOWN_TOOL
Macro

Syntax

```
#define PRL_ERR_STAND_BY_VM_WITH_UNSUPPORTED_SHUTDOWN_TOOL PRL_RESULT_DECLARE_ERROR(36018)
```

File

PrlErrorsValues.h

PRL_ERR_START_SAFE_MODE_UNSUPPORTED Macro

Syntax

```
#define PRL_ERR_START_SAFE_MODE_UNSUPPORTED PRL_RESULT_DECLARE_ERROR(547)
```

File

PrlErrorsValues.h

PRL_ERR_START_VM_BY_NOT_AUTH_USER Macro

Syntax

```
#define PRL_ERR_START_VM_BY_NOT_AUTH_USER PRL_RESULT_DECLARE_ERROR(530)
```

File

PrlErrorsValues.h

PRL_ERR_START_VM_BY_NOT_DEFINED_USER Macro

Syntax

```
#define PRL_ERR_START_VM_BY_NOT_DEFINED_USER PRL_RESULT_DECLARE_ERROR(529)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_STATE_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(19015)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_CANT_LOAD_SPECIFIED_CFG Macro

Syntax

```
#define PRL_ERR_STATE_CANT_LOAD_SPECIFIED_CFG PRL_RESULT_DECLARE_ERROR(19024)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_CANT_OPEN_FOR_WRITE Macro

Syntax

```
#define PRL_ERR_STATE_CANT_OPEN_FOR_WRITE PRL_RESULT_DECLARE_ERROR(19010)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_CANT_OPEN_IMAGE Macro

Syntax

```
#define PRL_ERR_STATE_CANT_OPEN_IMAGE PRL_RESULT_DECLARE_ERROR(19009)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_CANT_OPEN_LOCKED Macro

Syntax

```
#define PRL_ERR_STATE_CANT_OPEN_LOCKED PRL_RESULT_DECLARE_ERROR(19011)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_CANT_SAVE_LOCKED Macro

Syntax

```
#define PRL_ERR_STATE_CANT_SAVE_LOCKED PRL_RESULT_DECLARE_ERROR(19012)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_DELETE_NONCLOSED Macro

Syntax

```
#define PRL_ERR_STATE_DELETE_NONCLOSED PRL_RESULT_DECLARE_ERROR(19023)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_ERROR_CREATE_IMAGE Macro

Syntax

```
#define PRL_ERR_STATE_ERROR_CREATE_IMAGE PRL_RESULT_DECLARE_ERROR(19007)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_ERROR_RENAMING_IMAGE Macro

Syntax

```
#define PRL_ERR_STATE_ERROR_RENAMING_IMAGE PRL_RESULT_DECLARE_ERROR(19008)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_FULL_DELETE_FAILED Macro

Syntax

```
#define PRL_ERR_STATE_FULL_DELETE_FAILED PRL_RESULT_DECLARE_ERROR(19018)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_GETFREESPACE_FAILED Macro

Syntax

```
#define PRL_ERR_STATE_GETFREESPACE_FAILED PRL_RESULT_DECLARE_ERROR(19025)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_INT_CORRUPTED Macro

Syntax

```
#define PRL_ERR_STATE_INT_CORRUPTED PRL_RESULT_DECLARE_ERROR(19013)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_INVALID_IMAGE_TYPE Macro

Syntax

```
#define PRL_ERR_STATE_INVALID_IMAGE_TYPE PRL_RESULT_DECLARE_ERROR(19022)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_INVALID_PARAMETERS Macro

Syntax

```
#define PRL_ERR_STATE_INVALID_PARAMETERS PRL_RESULT_DECLARE_ERROR(19006)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_MEMORY_ERROR Macro

Syntax

```
#define PRL_ERR_STATE_MEMORY_ERROR PRL_RESULT_DECLARE_ERROR(19021)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_MERGE_FAILED Macro

Syntax

```
#define PRL_ERR_STATE_MERGE_FAILED PRL_RESULT_DECLARE_ERROR(19019)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_MERGE_NO_SPACE Macro

Syntax

```
#define PRL_ERR_STATE_MERGE_NO_SPACE PRL_RESULT_DECLARE_ERROR(19020)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_NO_DISKS Macro

Syntax

```
#define PRL_ERR_STATE_NO_DISKS PRL_RESULT_DECLARE_ERROR(19001)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_NO_STATE Macro

Syntax

```
#define PRL_ERR_STATE_NO_STATE PRL_RESULT_DECLARE_ERROR(19014)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_NOT_OPENED Macro

Syntax

```
#define PRL_ERR_STATE_NOT_OPENED PRL_RESULT_DECLARE_ERROR(19016)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_NOT_PERMITTED Macro

Syntax

```
#define PRL_ERR_STATE_NOT_PERMITTED PRL_RESULT_DECLARE_ERROR(19017)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_NOT_RELEASED Macro

Syntax

```
#define PRL_ERR_STATE_NOT_RELEASED PRL_RESULT_DECLARE_ERROR(19027)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_PROCESS_RUNNING Macro

Syntax

```
#define PRL_ERR_STATE_PROCESS_RUNNING PRL_RESULT_DECLARE_ERROR(19002)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_ROLLBACK_ERROR Macro

Syntax

```
#define PRL_ERR_STATE_ROLLBACK_ERROR PRL_RESULT_DECLARE_ERROR(19005)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_ROLLBACK_IN_PROGRESS Macro

Syntax

```
#define PRL_ERR_STATE_ROLLBACK_IN_PROGRESS PRL_RESULT_DECLARE_ERROR(19004)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_STATFS_FAILED Macro

Syntax

```
#define PRL_ERR_STATE_STATFS_FAILED PRL_RESULT_DECLARE_ERROR(19026)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_STOPPING_STATE Macro

Syntax

```
#define PRL_ERR_STATE_STOPPING_STATE PRL_RESULT_DECLARE_ERROR(19003)
```

File

PrlErrorsValues.h

PRL_ERR_STATE_UNEXPECTED_ERROR Macro

- Range reserved for different states error

+++ ALIGNED VALUE TO 19000

Syntax

```
#define PRL_ERR_STATE_UNEXPECTED_ERROR PRL_RESULT_DECLARE_ERROR(19000)
```

File

PrlErrorsValues.h

PRL_ERR_SUCCESS Macro

All Virtuozzo error values should go here in order to be maintained in one place We're taking approach of HRESULT windows notation error codes here.

Syntax

```
#define PRL_ERR_SUCCESS PRL_RESULT_DECLARE_SUCCESS(0)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_BOOTCAMP_NOT_NTFS_ONLY_DISK Macro

Syntax

```
#define PRL_ERR_SUSPEND_BOOTCAMP_NOT_NTFS_ONLY_DISK PRL_RESULT_DECLARE_ERROR(20015)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_BOOTCAMP_NTFS_RW_MOUNTERS_DETECTED Macro

Syntax

```
#define PRL_ERR_SUSPEND_BOOTCAMP_NTFS_RW_MOUNTERS_DETECTED PRL_RESULT_DECLARE_ERROR(20016)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_REJECTED_BY_GUEST Macro

Syntax

```
#define PRL_ERR_SUSPEND_REJECTED_BY_GUEST PRL_RESULT_DECLARE_ERROR(480)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_VM_VTD_BY_GUEST_SLEEP_TIMEOUT Macro

Syntax

```
#define PRL_ERR_SUSPEND_VM_VTD_BY_GUEST_SLEEP_TIMEOUT PRL_RESULT_DECLARE_ERROR(36002)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_VM_VTD_PAUSED Macro

VM error codes +++ ALIGNED VALUE TO 36000

Syntax

```
#define PRL_ERR_SUSPEND_VM_VTD_PAUSED PRL_RESULT_DECLARE_ERROR(36000)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_SUSPEND_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL PRL_RESULT_DECLARE_ERROR(36007)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_SUSPEND_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL PRL_RESULT_DECLARE_ERROR(36006)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_CAPS Macro

Syntax

```
#define PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_CAPS PRL_RESULT_DECLARE_ERROR(36025)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL Macro

Syntax

```
#define PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL PRL_RESULT_DECLARE_ERROR(36001)
```

File

PrlErrorsValues.h

PRL_ERR_SUSPEND_WITH_USB_BOOTDISK Macro

Syntax

```
#define PRL_ERR_SUSPEND_WITH_USB_BOOTDISK PRL_RESULT_DECLARE_ERROR(36029)
```

File

PrlErrorsValues.h

PRL_ERR_SYMBOL_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_SYMBOL_NOT_FOUND PRL_RESULT_DECLARE_ERROR(12)
```

File

PrlErrorsValues.h

PRL_ERR_TARGET_NAME_ALREADY_OCCUPIED Macro

Syntax

```
#define PRL_ERR_TARGET_NAME_ALREADY_OCCUPIED PRL_RESULT_DECLARE_ERROR(77)
```

File

PrlErrorsValues.h

PRL_ERR_TARGET_PATH_IS_NOT_DIRECTORY Macro

Syntax

```
#define PRL_ERR_TARGET_PATH_IS_NOT_DIRECTORY PRL_RESULT_DECLARE_ERROR(76)
```

File

PrlErrorsValues.h

PRL_ERR_TASK_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_TASK_NOT_FOUND PRL_RESULT_DECLARE_ERROR(92)
```

File

PrlErrorsValues.h

PRL_ERR_TEMPLATE_HAS_APPS Macro

containers templates errors +++ ALIGNED VALUE TO 46400

Syntax

```
#define PRL_ERR_TEMPLATE_HAS_APPS PRL_RESULT_DECLARE_ERROR(46400)
```

File

PrlErrorsValues.h

PRL_ERR_TEMPLATE_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_TEMPLATE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(46401)
```

File

PrlErrorsValues.h

PRL_ERR_TEST_TEXT_MESSAGES Macro

Test errors +++ ALIGNED VALUE TO 43200

Syntax

```
#define PRL_ERR_TEST_TEXT_MESSAGES PRL_RESULT_DECLARE_ERROR(43200)
```

File

PrlErrorsValues.h

PRL_ERR_TEST_TEXT_MESSAGES_PS Macro

Syntax

```
#define PRL_ERR_TEST_TEXT_MESSAGES_PS PRL_RESULT_DECLARE_ERROR(43201)
```

File

PrlErrorsValues.h

PRL_ERR_TIME_MACHINE_EXCLUDED_LIST_OP Macro

Syntax

```
#define PRL_ERR_TIME_MACHINE_EXCLUDED_LIST_OP PRL_RESULT_DECLARE_ERROR(546)
```

File

PrlErrorsValues.h

PRL_ERR_TIMEOUT Macro

Syntax

```
#define PRL_ERR_TIMEOUT PRL_RESULT_DECLARE_ERROR(15)
```

File

PrlErrorsValues.h

PRL_ERR_TIS_INVALID_UID Macro

- TIS errors

+++ ALIGNED VALUE TO 27800

Syntax

```
#define PRL_ERR_TIS_INVALID_UID PRL_RESULT_DECLARE_ERROR(27800)
```

File

PrlErrorsValues.h

PRL_ERR_TOO_LOW_HDD_SIZE Macro

Syntax

```
#define PRL_ERR_TOO_LOW_HDD_SIZE PRL_RESULT_DECLARE_ERROR(260)
```

File

PrlErrorsValues.h

PRL_ERR_TOOLS_UNSUPPORTED_GUEST Macro

Syntax

```
#define PRL_ERR_TOOLS_UNSUPPORTED_GUEST PRL_RESULT_DECLARE_ERROR(370)
```

File

PrlErrorsValues.h

PRL_ERR_TRY_AGAIN Macro

Syntax

```
#define PRL_ERR_TRY_AGAIN PRL_RESULT_DECLARE_ERROR(450)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_APPLY_MEMORY_GUARANTEE Macro

Syntax

```
#define PRL_ERR_UNABLE_APPLY_MEMORY_GUARANTEE PRL_RESULT_DECLARE_ERROR(538)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_APPLY_TOTAL_LIMIT Macro

Syntax

```
#define PRL_ERR_UNABLE_APPLY_TOTAL_LIMIT PRL_RESULT_DECLARE_ERROR(541)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_DROP_SUSPENDED_STATE Macro

Syntax

```
#define PRL_ERR_UNABLE_DROP_SUSPENDED_STATE PRL_RESULT_DECLARE_ERROR(103)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_SEND_REQUEST Macro

Syntax

```
#define PRL_ERR_UNABLE_SEND_REQUEST PRL_RESULT_DECLARE_ERROR(172)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_TO_CONTINUE_VM_LIFETIME_IS_EXPIRED Macro

Syntax

```
#define PRL_ERR_UNABLE_TO_CONTINUE_VM_LIFETIME_IS_EXPIRED PRL_RESULT_DECLARE_ERROR(573)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_TO_SEND_REQUEST Macro

Syntax

```
#define PRL_ERR_UNABLE_TO_SEND_REQUEST PRL_RESULT_DECLARE_ERROR(11714)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE Macro

Syntax

```
#define PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE PRL_RESULT_DECLARE_ERROR(566)
```

File

PrlErrorsValues.h

PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE_BY_NON_PRIVILEGED_USER Macro

Syntax

```
#define PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE_BY_NON_PRIVILEGED_USER
PRL_RESULT_DECLARE_ERROR(567)
```

File

PrlErrorsValues.h

PRL_ERR_UNATTENDED_UNSUPPORTED_GUEST Macro

Syntax

```
#define PRL_ERR_UNATTENDED_UNSUPPORTED_GUEST PRL_RESULT_DECLARE_ERROR(40001)
```

File

PrlErrorsValues.h

PRL_ERR_UNDER_OLD_HYPERVISOR Macro

Syntax

```
#define PRL_ERR_UNDER_OLD_HYPERVISOR PRL_RESULT_DECLARE_ERROR(258)
```

File

PrlErrorsValues.h

PRL_ERR_UNEXPECTED Macro

Syntax

```
#define PRL_ERR_UNEXPECTED PRL_RESULT_DECLARE_ERROR(1)
```

File

PrlErrorsValues.h

PRL_ERR_UNIMPLEMENTED Macro

Syntax

```
#define PRL_ERR_UNIMPLEMENTED PRL_RESULT_DECLARE_ERROR(8)
```

File

PrlErrorsValues.h

PRL_ERR_UNINITIALIZED Macro

Syntax

```
#define PRL_ERR_UNINITIALIZED PRL_RESULT_DECLARE_ERROR(7)
```

File

PrlErrorsValues.h

PRL_ERR_UNNAMED_CT_MOVE Macro

Syntax

```
#define PRL_ERR_UNNAMED_CT_MOVE PRL_RESULT_DECLARE_ERROR(44009)
```

File

PrlErrorsValues.h

PRL_ERR_UNRECOGNIZED_REQUEST Macro

Syntax

```
#define PRL_ERR_UNRECOGNIZED_REQUEST PRL_RESULT_DECLARE_ERROR(91)
```

File

PrlErrorsValues.h

PRL_ERR_UNSUPPORTED_DEVICE_TYPE Macro

Syntax

```
#define PRL_ERR_UNSUPPORTED_DEVICE_TYPE PRL_RESULT_DECLARE_ERROR(1002)
```

File

PrlErrorsValues.h

PRL_ERR_UNSUPPORTED_FILE_SYSTEM Macro

Syntax

```
#define PRL_ERR_UNSUPPORTED_FILE_SYSTEM PRL_ERR_UNSUPPORTED_NETWORK_FILE_SYSTEM
```

File

PrlErrorsValues.h

PRL_ERR_UNSUPPORTED_LAYOUTS_STRUCTURE Macro

Syntax

```
#define PRL_ERR_UNSUPPORTED_LAYOUTS_STRUCTURE PRL_RESULT_DECLARE_ERROR(45008)
```

File

PrlErrorsValues.h

PRL_ERR_UNSUPPORTED_NETWORK_FILE_SYSTEM Macro

Syntax

```
#define PRL_ERR_UNSUPPORTED_NETWORK_FILE_SYSTEM PRL_RESULT_DECLARE_ERROR(399)
```

File

PrlErrorsValues.h

PRL_ERR_UNSUPPORTED_VIRTUAL_SOURCE Macro

VI convert errors +++ ALIGNED VALUE TO 45000

Syntax

```
#define PRL_ERR_UNSUPPORTED_VIRTUAL_SOURCE PRL_RESULT_DECLARE_ERROR(45000)
```

File

PrlErrorsValues.h

PRL_ERR_UPD_TOOLS_VER_VM_NOT_CONFIGURED Macro

Syntax

```
#define PRL_ERR_UPD_TOOLS_VER_VM_NOT_CONFIGURED PRL_RESULT_DECLARE_ERROR(174)
```

File

PrlErrorsValues.h

PRL_ERR_UPD_TOOLS_VER_VM_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_UPD_TOOLS_VER_VM_NOT_CREATED PRL_RESULT_DECLARE_ERROR(173)
```

File

PrlErrorsValues.h

PRL_ERR_UPD_UPDATER_CONFIG Macro

- Range reserved for updater errors

+++ ALIGNED VALUE TO 17000

Syntax

```
#define PRL_ERR_UPD_UPDATER_CONFIG PRL_RESULT_DECLARE_ERROR(17000)
```

File

PrlErrorsValues.h

PRL_ERR_UPD_UPDATES Macro

Syntax

```
#define PRL_ERR_UPD_UPDATES PRL_RESULT_DECLARE_ERROR(17001)
```

File

PrlErrorsValues.h

PRL_ERR_UPDATE_MEM_VM_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_UPDATE_MEM_VM_NOT_CREATED PRL_RESULT_DECLARE_ERROR(171)
```

File

PrlErrorsValues.h

PRL_ERR_USER_CANT_CHANGE_ACCESS_PROFILE Macro

Syntax

```
#define PRL_ERR_USER_CANT_CHANGE_ACCESS_PROFILE PRL_RESULT_DECLARE_ERROR(119)
```

File

PrlErrorsValues.h

PRL_ERR_USER_CANT_CHANGE_PROFILE_ACCESS_PART Macro

Syntax

```
#define PRL_ERR_USER_CANT_CHANGE_PROFILE_ACCESS_PART PRL_RESULT_DECLARE_ERROR(259)
```

File

PrlErrorsValues.h

PRL_ERR_USER_CANT_CHANGE_READ_ONLY_VALUE Macro

Syntax

```
#define PRL_ERR_USER_CANT_CHANGE_READ_ONLY_VALUE PRL_RESULT_DECLARE_ERROR(120)
```

File

PrlErrorsValues.h

PRL_ERR_USER_DIRECTORY_NOT_SET Macro

Syntax

```
#define PRL_ERR_USER_DIRECTORY_NOT_SET PRL_RESULT_DECLARE_ERROR(67)
```

File

PrlErrorsValues.h

PRL_ERR_USER_IS_ALREADY_LOGGED Macro

Syntax

```
#define PRL_ERR_USER_IS_ALREADY_LOGGED PRL_RESULT_DECLARE_ERROR(61)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NO_AUTH_TO_CREATE_ROOT_VM_DIR Macro

Syntax

```
#define PRL_ERR_USER_NO_AUTH_TO_CREATE_ROOT_VM_DIR PRL_RESULT_DECLARE_ERROR(468)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NO_AUTH_TO_CREATE_VM_IN_DIR Macro

Syntax

```
#define PRL_ERR_USER_NO_AUTH_TO_CREATE_VM_IN_DIR PRL_RESULT_DECLARE_ERROR(316)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NO_AUTH_TO_EDIT_SERVER_SETTINGS Macro

Syntax

```
#define PRL_ERR_USER_NO_AUTH_TO_EDIT_SERVER_SETTINGS PRL_RESULT_DECLARE_ERROR(257)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NO_AUTH_TO_EDIT_VM Macro

Syntax

```
#define PRL_ERR_USER_NO_AUTH_TO_EDIT_VM PRL_RESULT_DECLARE_ERROR(321)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NO_AUTH_TO_SAVE_BACKUP_FILES Macro

Syntax

```
#define PRL_ERR_USER_NO_AUTH_TO_SAVE_BACKUP_FILES PRL_RESULT_DECLARE_ERROR(37030)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NO_AUTH_TO_SAVE_FILES Macro

Syntax

```
#define PRL_ERR_USER_NO_AUTH_TO_SAVE_FILES PRL_RESULT_DECLARE_ERROR(313)
```

File

PrlErrorsValues.h

PRL_ERR_USER_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_USER_NOT_FOUND PRL_RESULT_DECLARE_ERROR(390)
```

File

PrlErrorsValues.h

PRL_ERR_USER_OPERATION_NOT_AUTHORISED Macro

Syntax

```
#define PRL_ERR_USER_OPERATION_NOT_AUTHORISED PRL_RESULT_DECLARE_ERROR(59)
```

File

PrlErrorsValues.h

PRL_ERR_USER_PROFILE_WAS_CHANGED Macro

Syntax

```
#define PRL_ERR_USER_PROFILE_WAS_CHANGED PRL_RESULT_DECLARE_ERROR(118)
```

File

PrlErrorsValues.h

PRL_ERR_VA_CONFIG Macro

- Virtual Application Configuration errors

+++ ALIGNED VALUE TO 18000

Syntax

```
#define PRL_ERR_VA_CONFIG PRL_RESULT_DECLARE_ERROR(18000)
```

File

PrlErrorsValues.h

PRL_ERR_VCMMD_NO_CONNECTION Macro

Syntax

```
#define PRL_ERR_VCMMD_NO_CONNECTION PRL_RESULT_DECLARE_ERROR(581)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ABORT Macro

Syntax

```
#define PRL_ERR_VM_ABORT PRL_RESULT_DECLARE_ERROR(325)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALLOC_MEM_DRV_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_VM_ALLOC_MEM_DRV_NOT_STARTED PRL_RESULT_DECLARE_ERROR(184)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALLOC_VM_MEMORY Macro

Syntax

```
#define PRL_ERR_VM_ALLOC_VM_MEMORY PRL_RESULT_DECLARE_ERROR(185)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_CONNECTED Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_CONNECTED PRL_RESULT_DECLARE_ERROR(391)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_CREATED Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_CREATED PRL_RESULT_DECLARE_ERROR(124)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_REGISTERED Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_REGISTERED PRL_RESULT_DECLARE_ERROR(447)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_REGISTERED_UNIQUE_PARAMS Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_REGISTERED_UNIQUE_PARAMS PRL_RESULT_DECLARE_ERROR(114)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_REGISTERED_VM_NAME Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_REGISTERED_VM_NAME PRL_RESULT_DECLARE_ERROR(115)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_REGISTERED_VM_PATH Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_REGISTERED_VM_PATH PRL_RESULT_DECLARE_ERROR(63)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_REGISTERED_VM_UUID Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_REGISTERED_VM_UUID PRL_RESULT_DECLARE_ERROR(62)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_RESUMED Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_RESUMED PRL_RESULT_DECLARE_ERROR(52)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_RUNNING PRL_RESULT_DECLARE_ERROR(402)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ALREADY_SUSPENDED Macro

Syntax

```
#define PRL_ERR_VM_ALREADY_SUSPENDED PRL_RESULT_DECLARE_ERROR(50)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ANOTHER_TOOLS_IN_USE Macro

Syntax

```
#define PRL_ERR_VM_ANOTHER_TOOLS_IN_USE PRL_RESULT_DECLARE_ERROR(425)
```

File

PrlErrorsValues.h

PRL_ERR_VM_APPLY_CHANGES_PENDING Macro

Syntax

```
#define PRL_ERR_VM_APPLY_CHANGES_PENDING PRL_RESULT_DECLARE_ERROR(366)
```

File

PrlErrorsValues.h

PRL_ERR_VM_APPLY_CONFIG_FAILED Macro

Syntax

```
#define PRL_ERR_VM_APPLY_CONFIG_FAILED PRL_RESULT_DECLARE_ERROR(290)
```

File

PrlErrorsValues.h

PRL_ERR_VM_APPLY_CONFIG_NEEDS_REBOOT Macro

Syntax

```
#define PRL_ERR_VM_APPLY_CONFIG_NEEDS_REBOOT PRL_RESULT_DECLARE_ERROR(291)
```

File

PrlErrorsValues.h

PRL_ERR_VM_ASYNC_CD_CONSTRUCTOR Macro

Syntax

```
#define PRL_ERR_VM_ASYNC_CD_CONSTRUCTOR PRL_RESULT_DECLARE_ERROR(180)
```

File

PrlErrorsValues.h

PRL_ERR_VM_AUTO_COMPRESS_TASK_ALREADY_RUN Macro

Syntax

```
#define PRL_ERR_VM_AUTO_COMPRESS_TASK_ALREADY_RUN PRL_RESULT_DECLARE_ERROR(497)
```

File

PrlErrorsValues.h

PRL_ERR_VM_BACKUP_HDD_IMAGE_OUT_OF_BUNDLE Macro

Syntax

```
#define PRL_ERR_VM_BACKUP_HDD_IMAGE_OUT_OF_BUNDLE PRL_RESULT_DECLARE_ERROR(37035)
```

File

PrlErrorsValues.h

PRL_ERR_VM_BACKUP_INVALID_DISK_TYPE Macro

Syntax

```
#define PRL_ERR_VM_BACKUP_INVALID_DISK_TYPE PRL_RESULT_DECLARE_ERROR(37036)
```

File

PrlErrorsValues.h

PRL_ERR_VM_BAD_OS_TYPE Macro

Syntax

```
#define PRL_ERR_VM_BAD_OS_TYPE PRL_RESULT_DECLARE_ERROR(202)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CANT_CLONE_HDD_MISSING Macro

Syntax

```
#define PRL_ERR_VM_CANT_CLONE_HDD_MISSING PRL_RESULT_DECLARE_ERROR(320)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CANT_CLONE_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_CANT_CLONE_RUNNING PRL_RESULT_DECLARE_ERROR(292)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CANT_DELETE_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_CANT_DELETE_RUNNING PRL_RESULT_DECLARE_ERROR(293)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CANT_UNREG_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_CANT_UNREG_RUNNING PRL_RESULT_DECLARE_ERROR(294)
```

File

PrlErrorsValues.h

PRL_ERR_VM_COMPACT_HARD_DISK_FAILED Macro

Syntax

```
#define PRL_ERR_VM_COMPACT_HARD_DISK_FAILED PRL_RESULT_DECLARE_ERROR(496)
```

File

PrlErrorsValues.h

PRL_ERR_VM_COMPACT_NOT_SUPPORTED_GUEST_OS Macro

Syntax

```
#define PRL_ERR_VM_COMPACT_NOT_SUPPORTED_GUEST_OS PRL_RESULT_DECLARE_ERROR(500)
```

File

PrlErrorsValues.h

PRL_ERR_VM_COMPACT_PROCESSING Macro

- Optimization of virtual HDD Devices error

+++ ALIGNED VALUE TO 28000

Syntax

```
#define PRL_ERR_VM_COMPACT_PROCESSING PRL_RESULT_DECLARE_ERROR(28000)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONF_CHANGED_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_CONF_CHANGED_INVALID_ARG PRL_RESULT_DECLARE_ERROR(134)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONF_CHANGED_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_CONF_CHANGED_NOT_CREATED PRL_RESULT_DECLARE_ERROR(133)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONF_CHANGED_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_VM_CONF_CHANGED_NOT_STARTED PRL_RESULT_DECLARE_ERROR(135)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONF_CHANGED_UNSUPPORTED_DEV Macro

Syntax

```
#define PRL_ERR_VM_CONF_CHANGED_UNSUPPORTED_DEV PRL_RESULT_DECLARE_ERROR(136)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(37)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_CAN_BE_RESTORED Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_CAN_BE_RESTORED PRL_RESULT_DECLARE_ERROR(454)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_DOESNT_EXIST Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_DOESNT_EXIST PRL_RESULT_DECLARE_ERROR(283)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_INVALID_SERVER_UUID Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_INVALID_SERVER_UUID PRL_RESULT_DECLARE_ERROR(375)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_INVALID_VM_UUID Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_INVALID_VM_UUID PRL_RESULT_DECLARE_ERROR(374)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_IS_ALREADY_VALID Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_IS_ALREADY_VALID PRL_RESULT_DECLARE_ERROR(455)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CONFIG_WAS_CHANGED Macro

Syntax

```
#define PRL_ERR_VM_CONFIG_WAS_CHANGED PRL_RESULT_DECLARE_ERROR(112)
```

File

PrlErrorsValues.h

PRL_ERR_VM_COULDNT_BE_STARTED_UNDER_SPECIFIED_USER Macro

Syntax

```
#define PRL_ERR_VM_COULDNT_BE_STARTED_UNDER_SPECIFIED_USER PRL_RESULT_DECLARE_ERROR(398)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CREATE_HDD_IMG_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_CREATE_HDD_IMG_INVALID_ARG PRL_RESULT_DECLARE_ERROR(167)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CREATE_HDD_IMG_INVALID_CREATE Macro

Syntax

```
#define PRL_ERR_VM_CREATE_HDD_IMG_INVALID_CREATE PRL_RESULT_DECLARE_ERROR(168)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CREATE_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_CREATE_INVALID_ARG PRL_RESULT_DECLARE_ERROR(123)
```

File

PrlErrorsValues.h

PRL_ERR_VM_CREATE_SNAPSHOT_FAILED Macro

Syntax

```
#define PRL_ERR_VM_CREATE_SNAPSHOT_FAILED PRL_ERR_VM_SAVE_STATE_FAILED
```

File

PrlErrorsValues.h

PRL_ERR_VM_DELETE_STATE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_DELETE_STATE_FAILED PRL_RESULT_DECLARE_ERROR(427)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DEV_CHANGE_MEDIA_FAILED Macro

Syntax

```
#define PRL_ERR_VM_DEV_CHANGE_MEDIA_FAILED PRL_RESULT_DECLARE_ERROR(57)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DEV_CONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_VM_DEV_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(55)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DEV_DISCONNECT_FAILED Macro

Syntax

```
#define PRL_ERR_VM_DEV_DISCONNECT_FAILED PRL_RESULT_DECLARE_ERROR(56)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DEVICES_INITIALIZATION_FAILED Macro

Syntax

```
#define PRL_ERR_VM_DEVICES_INITIALIZATION_FAILED PRL_RESULT_DECLARE_ERROR(273)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DEVICES_TERMINATION_FAILED Macro

Syntax

```
#define PRL_ERR_VM_DEVICES_TERMINATION_FAILED PRL_RESULT_DECLARE_ERROR(274)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DIR_CONFIG_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_VM_DIR_CONFIG_ALREADY_EXISTS PRL_RESULT_DECLARE_ERROR(31)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DIR_FILE_NOT_SET Macro

Syntax

```
#define PRL_ERR_VM_DIR_FILE_NOT_SET PRL_RESULT_DECLARE_ERROR(39)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DIRECTORY_FOLDER_DOESNT_EXIST Macro

Syntax

```
#define PRL_ERR_VM_DIRECTORY_FOLDER_DOESNT_EXIST PRL_RESULT_DECLARE_ERROR(449)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DIRECTORY_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_VM_DIRECTORY_NOT_EXIST PRL_RESULT_DECLARE_ERROR(327)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DIRECTORY_NOT_INITIALIZED Macro

Syntax

```
#define PRL_ERR_VM_DIRECTORY_NOT_INITIALIZED PRL_RESULT_DECLARE_ERROR(65)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DOES_NOT_STOPPED Macro

Syntax

```
#define PRL_ERR_VM_DOES_NOT_STOPPED PRL_RESULT_DECLARE_ERROR(241)
```

File

PrlErrorsValues.h

PRL_ERR_VM_DVD_DISCONNECT_FAILED_LOCKED Macro

Syntax

```
#define PRL_ERR_VM_DVD_DISCONNECT_FAILED_LOCKED PRL_RESULT_DECLARE_ERROR(434)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EDIT_UNABLE_CHANGE_IP_ADDRESS Macro

Syntax

```
#define PRL_ERR_VM_EDIT_UNABLE_CHANGE_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(457)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EDIT_UNABLE_CONVERT_TO_TEMPLATE Macro

Syntax

```
#define PRL_ERR_VM_EDIT_UNABLE_CONVERT_TO_TEMPLATE PRL_RESULT_DECLARE_ERROR(576)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EDIT_UNABLE_SWITCH_OFF_UNDO_DISKS_MODE Macro

Syntax

```
#define PRL_ERR_VM_EDIT_UNABLE_SWITCH_OFF_UNDO_DISKS_MODE PRL_RESULT_DECLARE_ERROR(451)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EDIT_UNABLE_SWITCH_ON_UNDO_DISKS_MODE Macro

Syntax

```
#define PRL_ERR_VM_EDIT_UNABLE_SWITCH_ON_UNDO_DISKS_MODE PRL_RESULT_DECLARE_ERROR(435)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EMPTY_NAME_OF_CLONE Macro

Syntax

```
#define PRL_ERR_VM_EMPTY_NAME_OF_CLONE PRL_RESULT_DECLARE_ERROR(231)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXEC_GUEST_TOOL_NOT_AVAILABLE Macro

Syntax

```
#define PRL_ERR_VM_EXEC_GUEST_TOOL_NOT_AVAILABLE PRL_RESULT_DECLARE_ERROR(34001)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXEC_PROGRAM_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_VM_EXEC_PROGRAM_NOT_FOUND PRL_RESULT_DECLARE_ERROR(34003)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXPIRATION_HOST_DATETIME_SKEWED Macro

Syntax

```
#define PRL_ERR_VM_EXPIRATION_HOST_DATETIME_SKEWED PRL_RESULT_DECLARE_ERROR(570)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXPIRATION_OFFLINE_GRACE_PERIOD_EXPIRED
Macro

Syntax

```
#define PRL_ERR_VM_EXPIRATION_OFFLINE_GRACE_PERIOD_EXPIRED PRL_RESULT_DECLARE_ERROR(569)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXPIRATION_TIME_CHECK_INTERVAL_OUT_OF_RANGE
Macro

Syntax

```
#define PRL_ERR_VM_EXPIRATION_TIME_CHECK_INTERVAL_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(568)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXPIRATION_VM_IS_ABOUT_TO_EXPIRE
Macro

Syntax

```
#define PRL_ERR_VM_EXPIRATION_VM_IS_ABOUT_TO_EXPIRE PRL_RESULT_DECLARE_ERROR(572)
```

File

PrlErrorsValues.h

PRL_ERR_VM_EXPIRATION_VM_IS_IN_OFFLINE_GRACE_PERIOD
Macro

Syntax

```
#define PRL_ERR_VM_EXPIRATION_VM_IS_IN_OFFLINE_GRACE_PERIOD PRL_RESULT_DECLARE_ERROR(571)
```

File

PrlErrorsValues.h

PRL_ERR_VM_FILES_ALREADY_REMOVED Macro

Syntax

```
#define PRL_ERR_VM_FILES_ALREADY_REMOVED PRL_RESULT_DECLARE_ERROR(305)
```

File

PrlErrorsValues.h

PRL_ERR_VM_FREE_VM_MEMORY Macro

Syntax

```
#define PRL_ERR_VM_FREE_VM_MEMORY PRL_RESULT_DECLARE_ERROR(186)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_CONFIG_FAILED Macro

Syntax

```
#define PRL_ERR_VM_GET_CONFIG_FAILED PRL_RESULT_DECLARE_ERROR(49)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_CONFIG_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_GET_CONFIG_INVALID_ARG PRL_RESULT_DECLARE_ERROR(130)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_CONFIG_NOT_CONFIGURED Macro

Syntax

```
#define PRL_ERR_VM_GET_CONFIG_NOT_CONFIGURED PRL_RESULT_DECLARE_ERROR(131)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_DEVICE_STATE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_GET_DEVICE_STATE_FAILED PRL_RESULT_DECLARE_ERROR(58)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_HDD_IMG_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_GET_HDD_IMG_INVALID_ARG PRL_RESULT_DECLARE_ERROR(169)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_HDD_IMG_NOT_OPEN Macro

Syntax

```
#define PRL_ERR_VM_GET_HDD_IMG_NOT_OPEN PRL_RESULT_DECLARE_ERROR(170)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_INDICATORS_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_GET_INDICATORS_INVALID_ARG PRL_RESULT_DECLARE_ERROR(165)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_INDICATORS_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_GET_INDICATORS_NOT_CREATED PRL_RESULT_DECLARE_ERROR(164)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_INDICATORS_NOT_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_GET_INDICATORS_NOT_RUNNING PRL_RESULT_DECLARE_ERROR(166)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_PROBLEM_REPORT_FAILED Macro

Syntax

```
#define PRL_ERR_VM_GET_PROBLEM_REPORT_FAILED PRL_RESULT_DECLARE_ERROR(48)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_SCR_SIZE_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_GET_SCR_SIZE_NOT_CREATED PRL_RESULT_DECLARE_ERROR(148)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_SCR_UPDATED_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_GET_SCR_UPDATED_NOT_CREATED PRL_RESULT_DECLARE_ERROR(149)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_SCR_UPDATED_NOT_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_GET_SCR_UPDATED_NOT_RUNNING PRL_RESULT_DECLARE_ERROR(150)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_STATUS_FAILED Macro

Syntax

```
#define PRL_ERR_VM_GET_STATUS_FAILED PRL_RESULT_DECLARE_ERROR(47)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GET_STATUS_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_GET_STATUS_INVALID_ARG PRL_RESULT_DECLARE_ERROR(126)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GUEST_SESSION_EXPIRED Macro

Syntax

```
#define PRL_ERR_VM_GUEST_SESSION_EXPIRED PRL_RESULT_DECLARE_ERROR(34002)
```

File

PrlErrorsValues.h

PRL_ERR_VM_GUESTMEM_FAIL Macro

Syntax

```
#define PRL_ERR_VM_GUESTMEM_FAIL PRL_RESULT_DECLARE_ERROR(392)
```

File

PrlErrorsValues.h

PRL_ERR_VM_HARD_DISK_NOT_COMPACTABLE Macro

Syntax

```
#define PRL_ERR_VM_HARD_DISK_NOT_COMPACTABLE PRL_RESULT_DECLARE_ERROR(498)
```

File

PrlErrorsValues.h

PRL_ERR_VM_HDD_DISCONNECT_FAILED_LOCKED Macro

Syntax

```
#define PRL_ERR_VM_HDD_DISCONNECT_FAILED_LOCKED PRL_RESULT_DECLARE_ERROR(528)
```

File

PrlErrorsValues.h

PRL_ERR_VM_HDD_SIZE Macro

Syntax

```
#define PRL_ERR_VM_HDD_SIZE PRL_RESULT_DECLARE_ERROR(177)
```

File

PrlErrorsValues.h

PRL_ERR_VM_HOME_PATH_IS_NOT_EMPTY Macro

Syntax

```
#define PRL_ERR_VM_HOME_PATH_IS_NOT_EMPTY PRL_RESULT_DECLARE_ERROR(228)
```

File

PrlErrorsValues.h

PRL_ERR_VM_HYPERVISOR_HVT_ENABLED_ON_HYPERSWITCH Macro

Syntax

```
#define PRL_ERR_VM_HYPERVISOR_HVT_ENABLED_ON_HYPERSWITCH PRL_RESULT_DECLARE_ERROR(559)
```

File

PrlErrorsValues.h

PRL_ERR_VM_HYPERVIZOR_COMM Macro

Syntax

```
#define PRL_ERR_VM_HYPERVIZOR_COMM PRL_RESULT_DECLARE_ERROR(198)
```

File

PrlErrorsValues.h

PRL_ERR_VM_IN_FROZEN_STATE Macro

Syntax

```
#define PRL_ERR_VM_IN_FROZEN_STATE PRL_RESULT_DECLARE_ERROR(554)
```

File

PrlErrorsValues.h

PRL_ERR_VM_IN_HDD_CONSTRUCTOR Macro

Syntax

```
#define PRL_ERR_VM_IN_HDD_CONSTRUCTOR PRL_RESULT_DECLARE_ERROR(178)
```

File

PrlErrorsValues.h

PRL_ERR_VM_IN_WAKING_UP_STATE Macro

Syntax

```
#define PRL_ERR_VM_IN_WAKING_UP_STATE PRL_RESULT_DECLARE_ERROR(553)
```

File

PrlErrorsValues.h

PRL_ERR_VM_INIT_MONITOR Macro

Syntax

```
#define PRL_ERR_VM_INIT_MONITOR PRL_RESULT_DECLARE_ERROR(190)
```

File

PrlErrorsValues.h

PRL_ERR_VM_INIT_VCPU Macro

Syntax

```
#define PRL_ERR_VM_INIT_VCPU PRL_RESULT_DECLARE_ERROR(191)
```

File

PrlErrorsValues.h

PRL_ERR_VM_INTERACT_PRLS_DRIVER Macro

Syntax

```
#define PRL_ERR_VM_INTERACT_PRLS_DRIVER PRL_RESULT_DECLARE_ERROR(187)
```

File

PrlErrorsValues.h

PRL_ERR_VM_INTERNAL_OS_ERROR Macro

Syntax

```
#define PRL_ERR_VM_INTERNAL_OS_ERROR PRL_RESULT_DECLARE_ERROR(269)
```

File

PrlErrorsValues.h

PRL_ERR_VM_INVALID_SWAP_REGION Macro

Syntax

```
#define PRL_ERR_VM_INVALID_SWAP_REGION PRL_RESULT_DECLARE_ERROR(378)
```

File

PrlErrorsValues.h

PRL_ERR_VM_IS_EXCLUSIVELY_LOCKED Macro

Syntax

```
#define PRL_ERR_VM_IS_EXCLUSIVELY_LOCKED PRL_RESULT_DECLARE_ERROR(476)
```

File

PrlErrorsValues.h

PRL_ERR_VM_IS_NOT_LOCKED Macro

Syntax

```
#define PRL_ERR_VM_IS_NOT_LOCKED PRL_RESULT_DECLARE_ERROR(477)
```

File

PrlErrorsValues.h

PRL_ERR_VM_IS_NOT_SUSPENDED Macro

Syntax

```
#define PRL_ERR_VM_IS_NOT_SUSPENDED PRL_RESULT_DECLARE_ERROR(420)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LIFETIME_IS_EXPIRED Macro

Syntax

```
#define PRL_ERR_VM_LIFETIME_IS_EXPIRED PRL_RESULT_DECLARE_ERROR(565)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOAD_BINARY_FAILED Macro

Syntax

```
#define PRL_ERR_VM_LOAD_BINARY_FAILED PRL_RESULT_DECLARE_ERROR(403)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOAD_MONITOR Macro

Syntax

```
#define PRL_ERR_VM_LOAD_MONITOR PRL_RESULT_DECLARE_ERROR(195)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_CTL_FOR_BACKUP Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_CTL_FOR_BACKUP PRL_RESULT_DECLARE_ERROR(38115)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_BACKUP Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_BACKUP PRL_RESULT_DECLARE_ERROR(38113)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_CHANGE_FIREWALL Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_CHANGE_FIREWALL PRL_RESULT_DECLARE_ERROR(38122)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_CHANGE_PASSWORD Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_CHANGE_PASSWORD PRL_RESULT_DECLARE_ERROR(38121)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_CLONE Macro

Dispatcher 'locked' error codes +++ ALIGNED VALUE TO 38100

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_CLONE PRL_RESULT_DECLARE_ERROR(38100)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_COPY_IMAGE Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_COPY_IMAGE PRL_RESULT_DECLARE_ERROR(38123)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_CREATE_SNAPSHOT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_CREATE_SNAPSHOT PRL_RESULT_DECLARE_ERROR(38110)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_DECRYPT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_DECRYPT PRL_RESULT_DECLARE_ERROR(38120)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_DELETE Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_DELETE PRL_RESULT_DECLARE_ERROR(38101)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_DELETE_TO_SNAPSHOT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_DELETE_TO_SNAPSHOT PRL_RESULT_DECLARE_ERROR(38112)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_DISK_COMPACT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_DISK_COMPACT PRL_RESULT_DECLARE_ERROR(38117)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_DISK_CONVERT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_DISK_CONVERT PRL_RESULT_DECLARE_ERROR(38118)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_DISK_RESIZE Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_DISK_RESIZE PRL_RESULT_DECLARE_ERROR(38116)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT PRL_RESULT_DECLARE_ERROR(38103)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT_WITH_RENAME Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT_WITH_RENAME PRL_RESULT_DECLARE_ERROR(38107)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_ENCRYPT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_ENCRYPT PRL_RESULT_DECLARE_ERROR(38119)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_EXECUTE Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_EXECUTE PRL_RESULT_DECLARE_ERROR(38104)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_EXECUTE_EX Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_EXECUTE_EX PRL_RESULT_DECLARE_ERROR(38105)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_INTERNAL_REASON Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_INTERNAL_REASON PRL_RESULT_DECLARE_ERROR(38106)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_MIGRATE Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_MIGRATE PRL_RESULT_DECLARE_ERROR(38109)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_MOVE Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_MOVE PRL_RESULT_DECLARE_ERROR(38124)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_REMOVE_PROTECTION Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_REMOVE_PROTECTION PRL_RESULT_DECLARE_ERROR(38126)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_RESTORE_FROM_BACKUP Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_RESTORE_FROM_BACKUP PRL_RESULT_DECLARE_ERROR(38114)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_SET_PROTECTION Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_SET_PROTECTION PRL_RESULT_DECLARE_ERROR(38125)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_SWITCH_TO_SNAPSHOT Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_SWITCH_TO_SNAPSHOT PRL_RESULT_DECLARE_ERROR(38111)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_UNREGISTER Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_UNREGISTER PRL_RESULT_DECLARE_ERROR(38102)
```

File

PrlErrorsValues.h

PRL_ERR_VM_LOCKED_FOR_UPDATE_SECURITY Macro

Syntax

```
#define PRL_ERR_VM_LOCKED_FOR_UPDATE_SECURITY PRL_RESULT_DECLARE_ERROR(38108)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MAP_VM_MEM0 Macro

Syntax

```
#define PRL_ERR_VM_MAP_VM_MEM0 PRL_RESULT_DECLARE_ERROR(189)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MEMORY_SWAPPING_IN_PROGRESS Macro

Syntax

```
#define PRL_ERR_VM_MEMORY_SWAPPING_IN_PROGRESS PRL_RESULT_DECLARE_ERROR(20009)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_ACCESS_TO_VM_DENIED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_ACCESS_TO_VM_DENIED PRL_RESULT_DECLARE_ERROR(31046)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_BREAK_BY_DISK_CONDITION Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_BREAK_BY_DISK_CONDITION PRL_RESULT_DECLARE_ERROR(31039)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_CANNOT_CREATE_DIRECTORY Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_CANNOT_CREATE_DIRECTORY PRL_RESULT_DECLARE_ERROR(31052)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_CANNOT_REMOTE_CLONE_SHARED_VM Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_CANNOT_REMOTE_CLONE_SHARED_VM PRL_RESULT_DECLARE_ERROR(31037)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_CHECKING_PRECONDITIONS_FAILED Macro

VM migration errors codes set +++ ALIGNED VALUE TO 31000

Syntax

```
#define PRL_ERR_VM_MIGRATE_CHECKING_PRECONDITIONS_FAILED PRL_RESULT_DECLARE_ERROR(31000)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_CONTINUE_START_FAILED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_CONTINUE_START_FAILED PRL_RESULT_DECLARE_ERROR(31038)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_COULDNT_DETACH_TARGET_CONNECTION Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_COULDNT_DETACH_TARGET_CONNECTION PRL_RESULT_DECLARE_ERROR(31016)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_DEVICE_IMAGE_OUT_OF_BUNDLE Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_DEVICE_IMAGE_OUT_OF_BUNDLE PRL_RESULT_DECLARE_ERROR(31041)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_ERROR_DELETE_VM Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_ERROR_DELETE_VM PRL_RESULT_DECLARE_ERROR(31047)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_ERROR_UNREGISTER_VM Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_ERROR_UNREGISTER_VM PRL_RESULT_DECLARE_ERROR(31048)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_EXT_DISK_DIR_ALREADY_EXISTS_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_EXT_DISK_DIR_ALREADY_EXISTS_ON_TARGET
PRL_RESULT_DECLARE_ERROR(31051)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_EXTERNAL_DISKS_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_EXTERNAL_DISKS_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(31050)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_FLOPPY_DISK_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_FLOPPY_DISK_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31007)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_INVALID_DISK_TYPE Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_INVALID_DISK_TYPE PRL_RESULT_DECLARE_ERROR(31042)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NETWORK_ADAPTER_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NETWORK_ADAPTER_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31011)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NETWORK_SHARE_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NETWORK_SHARE_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31006)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET PRL_RESULT_DECLARE_ERROR(31005)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET_SHORT
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET_SHORT PRL_RESULT_DECLARE_ERROR(31043)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_CPUS_ON_TARGET
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NOT_ENOUGH_CPUS_ON_TARGET PRL_RESULT_DECLARE_ERROR(31004)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_SOURCE
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_SOURCE PRL_RESULT_DECLARE_ERROR(31001)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_TARGET
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_TARGET PRL_RESULT_DECLARE_ERROR(31015)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_OPTICAL_DISK_IS_ABSENT_ON_TARGET
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_OPTICAL_DISK_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31008)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_OUT_OF_MEMORY_ON_TARGET
Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_OUT_OF_MEMORY_ON_TARGET PRL_RESULT_DECLARE_ERROR(31002)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_PARALLEL_PORT_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_PARALLEL_PORT_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31010)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_REGISTER_VM_FAILED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_REGISTER_VM_FAILED PRL_RESULT_DECLARE_ERROR(31031)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_REMOTE_DEVICE_IS_ATTACHED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_REMOTE_DEVICE_IS_ATTACHED PRL_RESULT_DECLARE_ERROR(31027)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_RESUME_FAILED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_RESUME_FAILED PRL_RESULT_DECLARE_ERROR(31029)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_RESUME_VM_FAILED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_RESUME_VM_FAILED PRL_RESULT_DECLARE_ERROR(31033)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_SERIAL_PORT_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_SERIAL_PORT_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31009)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_SOUND_DEVICE_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_SOUND_DEVICE_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31013)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_STORAGE_INFO_PARSE Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_STORAGE_INFO_PARSE PRL_RESULT_DECLARE_ERROR(31030)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_SUSPEND_VM_FAILED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_SUSPEND_VM_FAILED PRL_RESULT_DECLARE_ERROR(31032)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_TARGET_INSIDE_SHARED_VM_PRIVATE Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_TARGET_INSIDE_SHARED_VM_PRIVATE PRL_RESULT_DECLARE_ERROR(31053)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_TARGET_VM_HOME_PATH_NOT_EXISTS Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_TARGET_VM_HOME_PATH_NOT_EXISTS PRL_RESULT_DECLARE_ERROR(31003)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_TO_THE_SAME_NODE Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_TO_THE_SAME_NODE PRL_RESULT_DECLARE_ERROR(31045)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_UNSUITABLE_VM_STATE Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_UNSUITABLE_VM_STATE PRL_RESULT_DECLARE_ERROR(31017)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_USB_CONTROLLER_IS_ABSENT_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_USB_CONTROLLER_IS_ABSENT_ON_TARGET PRL_RESULT_DECLARE_ERROR(31012)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_VM_ALREADY_EXISTS_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_VM_ALREADY_EXISTS_ON_TARGET PRL_RESULT_DECLARE_ERROR(31014)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_VM_ALREADY_MIGRATE_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_VM_ALREADY_MIGRATE_ON_TARGET PRL_RESULT_DECLARE_ERROR(31034)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_VM_HOME_ALREADY_EXISTS_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_VM_HOME_ALREADY_EXISTS_ON_TARGET PRL_RESULT_DECLARE_ERROR(31026)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_VM_UUID_ALREADY_EXISTS_ON_TARGET Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_VM_UUID_ALREADY_EXISTS_ON_TARGET PRL_RESULT_DECLARE_ERROR(31025)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MIGRATE_WARM_MODE_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VM_MIGRATE_WARM_MODE_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(31040)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MONITOR Macro

Syntax

```
#define PRL_ERR_VM_MONITOR PRL_RESULT_DECLARE_ERROR(200)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MOUNT Macro

VM hdd mount/umount +++ ALIGNED VALUE TO 46600

Syntax

```
#define PRL_ERR_VM_MOUNT PRL_RESULT_DECLARE_ERROR(46600)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MUST_BE_STOPPED_BEFORE_RENAMING Macro

Syntax

```
#define PRL_ERR_VM_MUST_BE_STOPPED_BEFORE_RENAMING PRL_RESULT_DECLARE_ERROR(376)
```

File

PrlErrorsValues.h

PRL_ERR_VM_MUST_BE_STOPPED_FOR_CHANGE_DEVICES Macro

Syntax

```
#define PRL_ERR_VM_MUST_BE_STOPPED_FOR_CHANGE_DEVICES PRL_RESULT_DECLARE_ERROR(384)
```

File

PrlErrorsValues.h

PRL_ERR_VM_NAME_IS_EMPTY Macro

Syntax

```
#define PRL_ERR_VM_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(116)
```

File

PrlErrorsValues.h

PRL_ERR_VM_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_NOT_CREATED PRL_RESULT_DECLARE_ERROR(125)
```

File

PrlErrorsValues.h

PRL_ERR_VM_OPERATION_FAILED Macro

Syntax

```
#define PRL_ERR_VM_OPERATION_FAILED PRL_RESULT_DECLARE_ERROR(69)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PAUSE_ALREADY_PAUSED Macro

Syntax

```
#define PRL_ERR_VM_PAUSE_ALREADY_PAUSED PRL_RESULT_DECLARE_ERROR(147)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PAUSE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_PAUSE_FAILED PRL_RESULT_DECLARE_ERROR(51)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PAUSE_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_PAUSE_NOT_CREATED PRL_RESULT_DECLARE_ERROR(145)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PAUSE_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_VM_PAUSE_NOT_STARTED PRL_RESULT_DECLARE_ERROR(146)
```

File

PrlErrorsValues.h

PRL_ERR_VM_POWER_OFF_FAILED Macro

Syntax

```
#define PRL_ERR_VM_POWER_OFF_FAILED PRL_RESULT_DECLARE_ERROR(45)
```

File

PrlErrorsValues.h

PRL_ERR_VM_POWER_ON_FAILED Macro

Syntax

```
#define PRL_ERR_VM_POWER_ON_FAILED PRL_RESULT_DECLARE_ERROR(44)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PRLS_DESCTOP_ALLOC_PHYS_MEM Macro

Syntax

```
#define PRL_ERR_VM_PRLS_DESCTOP_ALLOC_PHYS_MEM PRL_RESULT_DECLARE_ERROR(199)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PROCESS_IS_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_VM_PROCESS_IS_NOT_STARTED PRL_RESULT_DECLARE_ERROR(68)
```

File

PrlErrorsValues.h

PRL_ERR_VM_PROCESS_IS_NOT_STARTED_BY_WRONG_HOST_SW Macro

Syntax

```
#define PRL_ERR_VM_PROCESS_IS_NOT_STARTED_BY_WRONG_HOST_SW PRL_RESULT_DECLARE_ERROR(502)
```

File

PrlErrorsValues.h

PRL_ERR_VM_REQUEST_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VM_REQUEST_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(201)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESET_FAILED Macro

Syntax

```
#define PRL_ERR_VM_RESET_FAILED PRL_RESULT_DECLARE_ERROR(46)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESTART_GUEST_FAILED Macro

Syntax

```
#define PRL_ERR_VM_RESTART_GUEST_FAILED PRL_RESULT_DECLARE_ERROR(438)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESTART_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_RESTART_NOT_CREATED PRL_RESULT_DECLARE_ERROR(143)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESTART_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_VM_RESTART_NOT_STARTED PRL_RESULT_DECLARE_ERROR(144)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESTORE_STATE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_RESTORE_STATE_FAILED PRL_RESULT_DECLARE_ERROR(54)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESTORE_STATE_FAILED_VM_STOPPED Macro

Syntax

```
#define PRL_ERR_VM_RESTORE_STATE_FAILED_VM_STOPPED PRL_RESULT_DECLARE_ERROR(503)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESUME_FAILED Macro

Syntax

```
#define PRL_ERR_VM_RESUME_FAILED PRL_RESULT_DECLARE_ERROR(20006)
```

File

PrlErrorsValues.h

PRL_ERR_VM_RESUME_INV_SAV_VERSION_CANCEL_RESUME Macro

Syntax

```
#define PRL_ERR_VM_RESUME_INV_SAV_VERSION_CANCEL_RESUME PRL_RESULT_DECLARE_SUCCESS(20007)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SAVE_STATE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_SAVE_STATE_FAILED PRL_RESULT_DECLARE_ERROR(53)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_ABS_MOUSE_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_SEND_ABS_MOUSE_NOT_CREATED PRL_RESULT_DECLARE_ERROR(158)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_ABS_MOUSE_NOT_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_SEND_ABS_MOUSE_NOT_RUNNING PRL_RESULT_DECLARE_ERROR(159)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_ABS_MOUSE_PAUSED Macro

Syntax

```
#define PRL_ERR_VM_SEND_ABS_MOUSE_PAUSED PRL_RESULT_DECLARE_ERROR(160)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_KEYBOARD_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_SEND_KEYBOARD_NOT_CREATED PRL_RESULT_DECLARE_ERROR(161)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_KEYBOARD_NOT_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_SEND_KEYBOARD_NOT_RUNNING PRL_RESULT_DECLARE_ERROR(162)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_KEYBOARD_PAUSED Macro

Syntax

```
#define PRL_ERR_VM_SEND_KEYBOARD_PAUSED PRL_RESULT_DECLARE_ERROR(163)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_REL_MOUSE_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_SEND_REL_MOUSE_NOT_CREATED PRL_RESULT_DECLARE_ERROR(155)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_REL_MOUSE_NOT_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_SEND_REL_MOUSE_NOT_RUNNING PRL_RESULT_DECLARE_ERROR(156)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SEND_REL_MOUSE_PAUSED Macro

Syntax

```
#define PRL_ERR_VM_SEND_REL_MOUSE_PAUSED PRL_RESULT_DECLARE_ERROR(157)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SET_CONFIG_FAILED Macro

Syntax

```
#define PRL_ERR_VM_SET_CONFIG_FAILED PRL_RESULT_DECLARE_ERROR(43)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SET_CONFIG_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_SET_CONFIG_INVALID_ARG PRL_RESULT_DECLARE_ERROR(132)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SET_LICENSE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_SET_LICENSE_FAILED PRL_RESULT_DECLARE_ERROR(42)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SET_VISIBLE_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_SET_VISIBLE_NOT_CREATED PRL_RESULT_DECLARE_ERROR(151)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SHUTDOWN_FAILED Macro

Syntax

```
#define PRL_ERR_VM_SHUTDOWN_FAILED PRL_RESULT_DECLARE_ERROR(36020)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SHUTDOWN_HIBERNATE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_SHUTDOWN_HIBERNATE_FAILED PRL_RESULT_DECLARE_ERROR(36024)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SHUTDOWN_HIBERNATE_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VM_SHUTDOWN_HIBERNATE_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(36022)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SHUTDOWN_MACHINE_LOCKED Macro

Syntax

```
#define PRL_ERR_VM_SHUTDOWN_MACHINE_LOCKED PRL_RESULT_DECLARE_ERROR(36021)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SHUTDOWN_SUSPEND_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VM_SHUTDOWN_SUSPEND_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(36023)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SNAPSHOT_CHANGED_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_VM_SNAPSHOT_CHANGED_VM_CONFIG PRL_RESULT_DECLARE_ERROR(428)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SNAPSHOT_IN_SAFE_MODE Macro

Syntax

```
#define PRL_ERR_VM_SNAPSHOT_IN_SAFE_MODE PRL_RESULT_DECLARE_ERROR(440)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SNAPSHOT_IN_UNDO_DISKS_MODE Macro

Syntax

```
#define PRL_ERR_VM_SNAPSHOT_IN_UNDO_DISKS_MODE PRL_RESULT_DECLARE_ERROR(429)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SNAPSHOT_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_VM_SNAPSHOT_NOT_FOUND PRL_RESULT_DECLARE_ERROR(472)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SNAPSHOTS_CONFIG_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_VM_SNAPSHOTS_CONFIG_NOT_FOUND PRL_RESULT_DECLARE_ERROR(473)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SPECIFY_GUEST_INSTALL_FILES Macro

Syntax

```
#define PRL_ERR_VM_SPECIFY_GUEST_INSTALL_FILES PRL_RESULT_DECLARE_ERROR(299)
```

File

PrlErrorsValues.h

PRL_ERR_VM_START_FAILED Macro

Syntax

```
#define PRL_ERR_VM_START_FAILED PRL_RESULT_DECLARE_ERROR(41)
```

File

PrlErrorsValues.h

PRL_ERR_VM_START_NOT_CONFIGURED Macro

Syntax

```
#define PRL_ERR_VM_START_NOT_CONFIGURED PRL_RESULT_DECLARE_ERROR(141)
```

File

PrlErrorsValues.h

PRL_ERR_VM_START_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_START_NOT_CREATED PRL_RESULT_DECLARE_ERROR(140)
```

File

PrlErrorsValues.h

PRL_ERR_VM_STOP_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_STOP_NOT_CREATED PRL_RESULT_DECLARE_ERROR(142)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SUSPEND_CHANGED_VM_CONFIG Macro

Syntax

```
#define PRL_ERR_VM_SUSPEND_CHANGED_VM_CONFIG PRL_RESULT_DECLARE_ERROR(20008)
```

File

PrlErrorsValues.h

PRL_ERR_VM_SUSPEND_FAILED Macro

Syntax

```
#define PRL_ERR_VM_SUSPEND_FAILED PRL_RESULT_DECLARE_ERROR(20005)
```

File

PrlErrorsValues.h

PRL_ERR_VM_TO_REGISTER_IS_NOT_SPECIFIED Macro

Syntax

```
#define PRL_ERR_VM_TO_REGISTER_IS_NOT_SPECIFIED PRL_RESULT_DECLARE_ERROR(337)
```

File

PrlErrorsValues.h

PRL_ERR_VM_TOOL_NOT_AVAILABLE Macro

Syntax

```
#define PRL_ERR_VM_TOOL_NOT_AVAILABLE PRL_RESULT_DECLARE_ERROR(33002)
```

File

PrlErrorsValues.h

PRL_ERR_VM_TOOLS_CANT_PARSE_UPDATE_PARAMETERS Macro

VM tools erros codes set +++ ALIGNED VALUE TO 33000

Syntax

```
#define PRL_ERR_VM_TOOLS_CANT_PARSE_UPDATE_PARAMETERS PRL_RESULT_DECLARE_ERROR(33000)
```

File

PrlErrorsValues.h

PRL_ERR_VM_TOOLS_CANT_UPDATE_WITHOUT_RESTART Macro

Syntax

```
#define PRL_ERR_VM_TOOLS_CANT_UPDATE_WITHOUT_RESTART PRL_RESULT_DECLARE_ERROR(33001)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_ALLOC_MEM Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_ALLOC_MEM PRL_RESULT_DECLARE_ERROR(188)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_ALLOC_MEM_MONITOR Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_ALLOC_MEM_MONITOR PRL_RESULT_DECLARE_ERROR(196)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_CREATE_TIMER Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_CREATE_TIMER PRL_RESULT_DECLARE_ERROR(197)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_GET_GUEST_CPU Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_GET_GUEST_CPU PRL_RESULT_DECLARE_ERROR(192)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_OPEN_DISK_IMAGE Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_OPEN_DISK_IMAGE PRL_RESULT_DECLARE_ERROR(181)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_SEND_REQUEST Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_SEND_REQUEST PRL_RESULT_DECLARE_ERROR(183)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNABLE_TO_OPEN_VIRTUAL_BRIDGE Macro

Syntax

```
#define PRL_ERR_VM_UNABLE_TO_OPEN_VIRTUAL_BRIDGE PRL_RESULT_DECLARE_ERROR(179)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNDEFINED_API_CALL Macro

Syntax

```
#define PRL_ERR_VM_UNDEFINED_API_CALL PRL_RESULT_DECLARE_ERROR(182)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNMOUNT Macro

Syntax

```
#define PRL_ERR_VM_UNMOUNT PRL_RESULT_DECLARE_ERROR(46601)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UNPAUSE_FAILED Macro

Syntax

```
#define PRL_ERR_VM_UNPAUSE_FAILED PRL_RESULT_DECLARE_ERROR(517)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UPDATE_SNAPSHOT_DATA_FAILED Macro

Syntax

```
#define PRL_ERR_VM_UPDATE_SNAPSHOT_DATA_FAILED PRL_RESULT_DECLARE_ERROR(432)
```

File

PrlErrorsValues.h

PRL_ERR_VM_USER_AUTHENTICATION_FAILED Macro

Syntax

```
#define PRL_ERR_VM_USER_AUTHENTICATION_FAILED PRL_RESULT_DECLARE_ERROR(396)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UUID_EMPTY Macro

Syntax

```
#define PRL_ERR_VM_UUID_EMPTY PRL_RESULT_DECLARE_ERROR(230)
```

File

PrlErrorsValues.h

PRL_ERR_VM_UUID_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_VM_UUID_NOT_FOUND PRL_RESULT_DECLARE_ERROR(105)
```

File

PrlErrorsValues.h

PRL_ERR_VM_VALLOC_MON_BODY Macro

Syntax

```
#define PRL_ERR_VM_VALLOC_MON_BODY PRL_RESULT_DECLARE_ERROR(193)
```

File

PrlErrorsValues.h

PRL_ERR_VM_VALLOC_PE_IMG Macro

Syntax

```
#define PRL_ERR_VM_VALLOC_PE_IMG PRL_RESULT_DECLARE_ERROR(194)
```

File

PrlErrorsValues.h

PRL_ERR_VM_VIEW_SCR_INVALID_ARG Macro

Syntax

```
#define PRL_ERR_VM_VIEW_SCR_INVALID_ARG PRL_RESULT_DECLARE_ERROR(153)
```

File

PrlErrorsValues.h

PRL_ERR_VM_VIEW_SCR_NOT_CREATED Macro

Syntax

```
#define PRL_ERR_VM_VIEW_SCR_NOT_CREATED PRL_RESULT_DECLARE_ERROR(152)
```

File

PrlErrorsValues.h

PRL_ERR_VM_VIEW_SCR_NOT_RUNNING Macro

Syntax

```
#define PRL_ERR_VM_VIEW_SCR_NOT_RUNNING PRL_RESULT_DECLARE_ERROR(154)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN
Macro

Unused PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN

Syntax

```
#define PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN PRL_RESULT_DECLARE_ERROR(27053)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOT_OPTION_DEVICE_NOT_EXISTS
Macro

Syntax

```
#define PRL_ERR_VMCONF_BOOT_OPTION_DEVICE_NOT_EXISTS PRL_RESULT_DECLARE_ERROR(27052)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOT_OPTION_DUPLICATE_DEVICE
Macro

Syntax

```
#define PRL_ERR_VMCONF_BOOT_OPTION_DUPLICATE_DEVICE PRL_RESULT_DECLARE_ERROR(27051)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOT_OPTION_INVALID_DEVICE_TYPE
Macro

ALIGNED VALUE TO 27050

Syntax

```
#define PRL_ERR_VMCONF_BOOT_OPTION_INVALID_DEVICE_TYPE PRL_RESULT_DECLARE_ERROR(27050)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOTCAMP_HARD_DISK_SMART_GUARD_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_BOOTCAMP_HARD_DISK_SMART_GUARD_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27016)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOTCAMP_HARD_SNAPSHOTS_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_BOOTCAMP_HARD_SNAPSHOTS_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27010)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOTCAMP_HARD_UNDO_DISKS_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_BOOTCAMP_HARD_UNDO_DISKS_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27008)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_BOOTCAMP_SAFE_MODE_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_BOOTCAMP_SAFE_MODE_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27012)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CD_DVD_ROM_DUPLICATE_SYS_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_CD_DVD_ROM_DUPLICATE_SYS_NAME PRL_RESULT_DECLARE_ERROR(27401)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CD_DVD_ROM_IMAGE_IS_NOT_EXIST
Macro

Syntax

```
#define PRL_ERR_VMCONF_CD_DVD_ROM_IMAGE_IS_NOT_EXIST PRL_RESULT_DECLARE_ERROR(27402)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_OLD_CHIPSET
Macro

Syntax

```
#define PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_OLD_CHIPSET PRL_RESULT_DECLARE_ERROR(27405)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_UNSUPPORTED_OS Macro

Syntax

```
#define PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_UNSUPPORTED_OS PRL_RESULT_DECLARE_ERROR(27404)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CD_DVD_ROM_SYS_NAME_IS_EMPTY Macro

ALIGNED VALUE TO 27400

Syntax

```
#define PRL_ERR_VMCONF_CD_DVD_ROM_SYS_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27400)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CD_DVD_ROM_URL_FORMAT_SYS_NAME Macro

Syntax

```
#define PRL_ERR_VMCONF_CD_DVD_ROM_URL_FORMAT_SYS_NAME PRL_RESULT_DECLARE_ERROR(27403)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPU_COUNT_MORE_HOST_CPU_COUNT Macro

Syntax

```
#define PRL_ERR_VMCONF_CPU_COUNT_MORE_HOST_CPU_COUNT PRL_RESULT_DECLARE_ERROR(27202)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPU_COUNT_MORE_MAX_CPU_COUNT Macro

Syntax

```
#define PRL_ERR_VMCONF_CPU_COUNT_MORE_MAX_CPU_COUNT PRL_RESULT_DECLARE_ERROR(27201)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPU_MASK_INVALID Macro

Syntax

```
#define PRL_ERR_VMCONF_CPU_MASK_INVALID PRL_RESULT_DECLARE_ERROR(27203)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPU_MASK_INVALID_CPU_NUM Macro

Syntax

```
#define PRL_ERR_VMCONF_CPU_MASK_INVALID_CPU_NUM PRL_RESULT_DECLARE_ERROR(27204)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPU_NODE_MASK_INVALID Macro

Syntax

```
#define PRL_ERR_VMCONF_CPU_NODE_MASK_INVALID PRL_RESULT_DECLARE_ERROR(27205)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPU_ZERO_COUNT Macro

ALIGNED VALUE TO 27200

Syntax

```
#define PRL_ERR_VMCONF_CPU_ZERO_COUNT PRL_RESULT_DECLARE_ERROR(27200)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPULIMIT_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VMCONF_CPULIMIT_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(27023)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_CPUUNITS_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VMCONF_CPUUNITS_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(27022)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_DESKTOP_MODE_REMOTE_DEVICES Macro

Syntax

```
#define PRL_ERR_VMCONF_DESKTOP_MODE_REMOTE_DEVICES PRL_RESULT_DECLARE_ERROR(27006)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_DUPLICATE_IP_ADDRESS Macro

Syntax

```
#define PRL_ERR_VMCONF_DUPLICATE_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27507)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_EFI_UNSUPPORTED_GUEST Macro

Syntax

```
#define PRL_ERR_VMCONF_EFI_UNSUPPORTED_GUEST PRL_RESULT_DECLARE_ERROR(27035)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_EXIST PRL_RESULT_DECLARE_ERROR(27351)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_VALID Macro

Syntax

```
#define PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_VALID PRL_RESULT_DECLARE_ERROR(27353)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_FLOPPY_DISK_IS_NOT_ACCESSIBLE Macro

Syntax

```
#define PRL_ERR_VMCONF_FLOPPY_DISK_IS_NOT_ACCESSIBLE PRL_RESULT_DECLARE_ERROR(27352)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_HAS_INVALID_SYMBOL Macro

Syntax

```
#define PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_HAS_INVALID_SYMBOL PRL_RESULT_DECLARE_ERROR(27354)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_IS_EMPTY Macro

ALIGNED VALUE TO 27350

Syntax

```
#define PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27350)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_FLOPPY_DISK_URL_FORMAT_SYS_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_FLOPPY_DISK_URL_FORMAT_SYS_NAME PRL_RESULT_DECLARE_ERROR(27355)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_CANNOT_BE_ADDED
Macro

ALIGNED VALUE TO 27760

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_CANNOT_BE_ADDED PRL_RESULT_DECLARE_ERROR(27760)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_DUPLICATE_IN_ANOTHER_VM
Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_DUPLICATE_IN_ANOTHER_VM
PRL_RESULT_DECLARE_ERROR(27764)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_CONNECTED
Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_CONNECTED PRL_RESULT_DECLARE_ERROR(27762)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_FOUND
Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(27761)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_DUPLICATE_SYS_NAME

Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_DUPLICATE_SYS_NAME PRL_RESULT_DECLARE_ERROR(27766)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE

Macro

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE is unused

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE PRL_RESULT_DECLARE_ERROR(27763)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_NOT_SINGLE Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_NOT_SINGLE PRL_RESULT_DECLARE_ERROR(27767)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_WRONG_COUNT Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_WRONG_COUNT
PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_NOT_SINGLE
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_GENERIC_PCI_WRONG_DEVICE Macro

Syntax

```
#define PRL_ERR_VMCONF_GENERIC_PCI_WRONG_DEVICE PRL_RESULT_DECLARE_ERROR(27765)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_DUPLICATE_SYS_NAME Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_DUPLICATE_SYS_NAME PRL_RESULT_DECLARE_ERROR(27453)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_EXIST PRL_RESULT_DECLARE_ERROR(27451)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_VALID Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_VALID PRL_RESULT_DECLARE_ERROR(27452)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_MISS_BOOTCAMP_PARTITION Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_MISS_BOOTCAMP_PARTITION PRL_RESULT_DECLARE_ERROR(27457)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_NOT_ENOUGH_SPACE_FOR_ENCRYPT_DISK
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_NOT_ENOUGH_SPACE_FOR_ENCRYPT_DISK
PRL_RESULT_DECLARE_ERROR(27459)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_SERIAL_IS_NOT_VALID
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_SERIAL_IS_NOT_VALID PRL_RESULT_DECLARE_ERROR(27463)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_OLD_CHIPSET
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_OLD_CHIPSET PRL_RESULT_DECLARE_ERROR(27461)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_UNSUPPORTED_OS
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_UNSUPPORTED_OS PRL_RESULT_DECLARE_ERROR(27460)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_HAS_INVALID_SYMBOL Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_HAS_INVALID_SYMBOL PRL_RESULT_DECLARE_ERROR(27454)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_IS_EMPTY Macro

ALIGNED VALUE TO 27450

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27450)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_UNABLE_DELETE_DISK_WITH_SNAPSHOTS Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_UNABLE_DELETE_DISK_WITH_SNAPSHOTS
PRL_RESULT_DECLARE_ERROR(27462)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_URL_FORMAT_SYS_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_URL_FORMAT_SYS_NAME PRL_RESULT_DECLARE_ERROR(27455)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_BOOTCAMP_PARTITION
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_BOOTCAMP_PARTITION PRL_RESULT_DECLARE_ERROR(27456)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_FOR_ENCRYPTRED_VM
Macro

Syntax

```
#define PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_FOR_ENCRYPTRED_VM PRL_RESULT_DECLARE_ERROR(27458)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_IDE_DEVICES_COUNT_OUT_OF_RANGE
Macro

ALIGNED VALUE TO 27700

Syntax

```
#define PRL_ERR_VMCONF_IDE_DEVICES_COUNT_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27700)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_IDE_DEVICES_DUPLICATE_STACK_INDEX
Macro

Syntax

```
#define PRL_ERR_VMCONF_IDE_DEVICES_DUPLICATE_STACK_INDEX PRL_RESULT_DECLARE_ERROR(27701)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_INCOMPAT_HARD_DISK_SMART_GUARD_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_INCOMPAT_HARD_DISK_SMART_GUARD_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27017)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_INCOMPAT_HARD_UNDO_DISKS_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_INCOMPAT_HARD_UNDO_DISKS_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27009)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_INCOMPAT_SAFE_MODE_NOT_ALLOW Macro

Syntax

```
#define PRL_ERR_VMCONF_INCOMPAT_SAFE_MODE_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27013)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_INVALID_DEVICE_MAIN_INDEX Macro

Syntax

```
#define PRL_ERR_VMCONF_INVALID_DEVICE_MAIN_INDEX PRL_RESULT_DECLARE_ERROR(27005)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_IOLIMIT_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VMCONF_IOLIMIT_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(27025)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_IOPRIO_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VMCONF_IOPRIO_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(27024)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_IOPSLIMIT_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VMCONF_IOPSLIMIT_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(27028)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_MAX_BALLOON_SIZE_MORE_100_PERCENT Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_MAX_BALLOON_SIZE_MORE_100_PERCENT
PRL_RESULT_DECLARE_ERROR(27254)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_INVALID_RANGE Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_MQ_INVALID_RANGE PRL_RESULT_DECLARE_ERROR(27257)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_LESS_VMM_OVERHEAD_VALUE Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_LESS_VMM_OVERHEAD_VALUE
PRL_RESULT_DECLARE_ERROR(27258)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_OUT_OF_RANGE
Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27259)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_OUT_OF_RANGE
Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27256)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_ZERO
Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_ZERO PRL_RESULT_DECLARE_ERROR(27255)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_NOT_4_RATIO_SIZE
Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_NOT_4_RATIO_SIZE PRL_RESULT_DECLARE_ERROR(27252)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_OUT_OF_RANGE Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27251)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_SIZE_NOT_EAQUAL_RECOMMENDED Macro

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_SIZE_NOT_EAQUAL_RECOMMENDED
PRL_RESULT_DECLARE_SUCCESS(27253)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_MAIN_MEMORY_ZERO_SIZE Macro

ALIGNED VALUE TO 27250

Syntax

```
#define PRL_ERR_VMCONF_MAIN_MEMORY_ZERO_SIZE PRL_RESULT_DECLARE_ERROR(27250)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTART Macro

Syntax

```
#define PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTART PRL_RESULT_DECLARE_ERROR(27030)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTOP
Macro

Syntax

```
#define PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTOP PRL_RESULT_DECLARE_ERROR(27031)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_KEEP_VM_ALIVE_ON_GUI_EXIT
Macro

Syntax

```
#define PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_KEEP_VM_ALIVE_ON_GUI_EXIT
PRL_RESULT_DECLARE_ERROR(27032)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NEED_MORE_MEMORY_TO_ENABLE_HOTPLUG
Macro

Syntax

```
#define PRL_ERR_VMCONF_NEED_MORE_MEMORY_TO_ENABLE_HOTPLUG PRL_RESULT_DECLARE_ERROR(27034)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_BROADCAST_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_BROADCAST_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27509)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27503)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_MAC_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_MAC_ADDRESS PRL_RESULT_DECLARE_ERROR(27502)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_ETHLIST_CREATE_ERROR
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_ETHLIST_CREATE_ERROR PRL_RESULT_DECLARE_ERROR(27504)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_GATEWAY_NOT_IN_SUBNET
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_GATEWAY_NOT_IN_SUBNET PRL_RESULT_DECLARE_ERROR(27511)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_GUEST_TOOLS_NOT_AVAILABLE
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_GUEST_TOOLS_NOT_AVAILABLE
PRL_RESULT_DECLARE_ERROR(27506)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_BOUND_INDEX
Macro

ALIGNED VALUE TO 27500

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_BOUND_INDEX PRL_RESULT_DECLARE_ERROR(27500)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_DNS_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_DNS_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27512)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_GATEWAY_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_GATEWAY_IP_ADDRESS
PRL_RESULT_DECLARE_ERROR(27510)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27505)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_MAC_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_MAC_ADDRESS PRL_RESULT_DECLARE_ERROR(27501)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_SEARCH_DOMAIN_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_SEARCH_DOMAIN_NAME
PRL_RESULT_DECLARE_ERROR(27513)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_MULTICAST_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_MULTICAST_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27508)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NETWORK_ADAPTER_ROUTED_NO_STATIC_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NETWORK_ADAPTER_ROUTED_NO_STATIC_ADDRESS PRL_RESULT_DECLARE_ERROR(27514)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_SMART_GUARD
Macro

Syntax

```
#define PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_SMART_GUARD PRL_RESULT_DECLARE_ERROR(27020)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_UNDO_DISKS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_UNDO_DISKS PRL_RESULT_DECLARE_ERROR(27019)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_CONFIGURED_TOTALRATE_FOR_NETWORK_CLASS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NO_CONFIGURED_TOTALRATE_FOR_NETWORK_CLASS
PRL_RESULT_DECLARE_ERROR(27772)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_HD_IMAGES_IN_SAFE_MODE
Macro

Syntax

```
#define PRL_ERR_VMCONF_NO_HD_IMAGES_IN_SAFE_MODE PRL_RESULT_DECLARE_ERROR(27015)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_HD_IMAGES_IN_UNDO_DISKS_MODE
Macro

Syntax

```
#define PRL_ERR_VMCONF_NO_HD_IMAGES_IN_UNDO_DISKS_MODE PRL_RESULT_DECLARE_ERROR(27014)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_IP_ADDRESSES_SPECIFIED_FOR_OFFLINE_MANAGEMENT
Macro

ALIGNED VALUE TO 27770

Syntax

```
#define PRL_ERR_VMCONF_NO_IP_ADDRESSES_SPECIFIED_FOR_OFFLINE_MANAGEMENT
PRL_RESULT_DECLARE_ERROR(27770)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_NO_SMART_GUARD_WITH_UNDO_DISKS
Macro

Syntax

```
#define PRL_ERR_VMCONF_NO_SMART_GUARD_WITH_UNDO_DISKS PRL_RESULT_DECLARE_ERROR(27021)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_PARALLEL_PORT_IMAGE_IS_NOT_EXIST
Macro

Syntax

```
#define PRL_ERR_VMCONF_PARALLEL_PORT_IMAGE_IS_NOT_EXIST PRL_RESULT_DECLARE_ERROR(27651)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
Macro

Syntax

```
#define PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
PRL_RESULT_DECLARE_ERROR(27652)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_IS_EMPTY
Macro

ALIGNED VALUE TO 27650

Syntax

```
#define PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27650)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_PARALLEL_PORT_URL_FORMAT_SYS_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_PARALLEL_PORT_URL_FORMAT_SYS_NAME PRL_RESULT_DECLARE_ERROR(27653)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REAL_HARD_SAFE_MODE_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_REAL_HARD_SAFE_MODE_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27011)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REAL_HARD_UNDO_DISKS_NOT_ALLOW
Macro

Syntax

```
#define PRL_ERR_VMCONF_REAL_HARD_UNDO_DISKS_NOT_ALLOW PRL_RESULT_DECLARE_ERROR(27007)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REMOTE_DISPLAY_EMPTY_PASSWORD
Macro

Syntax

```
#define PRL_ERR_VMCONF_REMOTE_DISPLAY_EMPTY_PASSWORD PRL_RESULT_DECLARE_ERROR(27103)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REMOTE_DISPLAY_HOST_IP_ADDRESS_IS_ZERO
Macro

Syntax

```
#define PRL_ERR_VMCONF_REMOTE_DISPLAY_HOST_IP_ADDRESS_IS_ZERO PRL_RESULT_DECLARE_ERROR(27101)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REMOTE_DISPLAY_INVALID_HOST_IP_ADDRESS
Macro

Syntax

```
#define PRL_ERR_VMCONF_REMOTE_DISPLAY_INVALID_HOST_IP_ADDRESS PRL_RESULT_DECLARE_ERROR(27102)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REMOTE_DISPLAY_PASSWORD_TOO_LONG
Macro

Syntax

```
#define PRL_ERR_VMCONF_REMOTE_DISPLAY_PASSWORD_TOO_LONG PRL_RESULT_DECLARE_ERROR(27104)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_REMOTE_DISPLAY_PORT_NUMBER_IS_ZERO
Macro

ALIGNED VALUE TO 27100

Syntax

```
#define PRL_ERR_VMCONF_REMOTE_DISPLAY_PORT_NUMBER_IS_ZERO PRL_RESULT_DECLARE_ERROR(27100)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_RESTRICTED_CPU_COUNT Macro

Syntax

```
#define PRL_ERR_VMCONF_RESTRICTED_CPU_COUNT PRL_RESULT_DECLARE_ERROR(26501)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_RESTRICTED_MEMORY_SIZE Macro

Syntax

```
#define PRL_ERR_VMCONF_RESTRICTED_MEMORY_SIZE PRL_RESULT_DECLARE_ERROR(26502)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_RESTRICTED_OS_VERSION Macro

- Validation VM configuration error

+++ ALIGNED VALUE TO 26500

Syntax

```
#define PRL_ERR_VMCONF_RESTRICTED_OS_VERSION PRL_RESULT_DECLARE_ERROR(26500)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_RESTRICTED_SMART_GUARD Macro

Syntax

```
#define PRL_ERR_VMCONF_RESTRICTED_SMART_GUARD PRL_RESULT_DECLARE_ERROR(26504)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_RESTRICTED_UNDO_DISKS Macro

Syntax

```
#define PRL_ERR_VMCONF_RESTRICTED_UNDO_DISKS PRL_RESULT_DECLARE_ERROR(26503)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SATA_DEVICES_COUNT_OUT_OF_RANGE Macro

Syntax

```
#define PRL_ERR_VMCONF_SATA_DEVICES_COUNT_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27702)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SATA_DEVICES_DUPLICATE_STACK_INDEX Macro

Syntax

```
#define PRL_ERR_VMCONF_SATA_DEVICES_DUPLICATE_STACK_INDEX PRL_RESULT_DECLARE_ERROR(27703)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SCSI_BUSLOGIC_WITH_EFI_NOT_SUPPORTED Macro

Syntax

```
#define PRL_ERR_VMCONF_SCSI_BUSLOGIC_WITH_EFI_NOT_SUPPORTED PRL_RESULT_DECLARE_ERROR(27752)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SCSI_DEVICES_COUNT_OUT_OF_RANGE
Macro

ALIGNED VALUE TO 27750

Syntax

```
#define PRL_ERR_VMCONF_SCSI_DEVICES_COUNT_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27750)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SCSI_DEVICES_DUPLICATE_STACK_INDEX
Macro

Syntax

```
#define PRL_ERR_VMCONF_SCSI_DEVICES_DUPLICATE_STACK_INDEX PRL_RESULT_DECLARE_ERROR(27751)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SERIAL_PORT_IMAGE_IS_NOT_EXIST
Macro

Syntax

```
#define PRL_ERR_VMCONF_SERIAL_PORT_IMAGE_IS_NOT_EXIST PRL_RESULT_DECLARE_ERROR(27601)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
Macro

Syntax

```
#define PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_HAS_INVALID_SYMBOL PRL_RESULT_DECLARE_ERROR(27602)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_IS_EMPTY
Macro

ALIGNED VALUE TO 27600

Syntax

```
#define PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27600)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SERIAL_PORT_URL_FORMAT_SYS_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_SERIAL_PORT_URL_FORMAT_SYS_NAME PRL_RESULT_DECLARE_ERROR(27603)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_NAME
Macro

Syntax

```
#define PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_NAME PRL_RESULT_DECLARE_ERROR(27151)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_PATH
Macro

Syntax

```
#define PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_PATH PRL_RESULT_DECLARE_ERROR(27153)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SHARED_FOLDERS_EMPTY_FOLDER_NAME
Macro

ALIGNED VALUE TO 27150

Syntax

```
#define PRL_ERR_VMCONF_SHARED_FOLDERS_EMPTY_FOLDER_NAME PRL_RESULT_DECLARE_ERROR(27150)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SHARED_FOLDERS_INVALID_FOLDER_PATH Macro

Syntax

```
#define PRL_ERR_VMCONF_SHARED_FOLDERS_INVALID_FOLDER_PATH PRL_RESULT_DECLARE_ERROR(27152)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY Macro

+++ ALIGNED VALUE TO 27550 +++ Unused PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY

Syntax

```
#define PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27550)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY Macro

Unused PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY

Syntax

```
#define PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27551)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_UNKNOWN_OS_TYPE Macro

Syntax

```
#define PRL_ERR_VMCONF_UNKNOWN_OS_TYPE PRL_RESULT_DECLARE_ERROR(27002)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_UNKNOWN_OS_VERSION Macro

Syntax

```
#define PRL_ERR_VMCONF_UNKNOWN_OS_VERSION PRL_RESULT_DECLARE_ERROR(27003)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VALIDATION_FAILED Macro

ALIGNED VALUE TO 27000

Syntax

```
#define PRL_ERR_VMCONF_VALIDATION_FAILED PRL_RESULT_DECLARE_ERROR(27000)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VIDEO_MEMORY_OUT_OF_RANGE Macro

ALIGNED VALUE TO 27300

Syntax

```
#define PRL_ERR_VMCONF_VIDEO_MEMORY_OUT_OF_RANGE PRL_RESULT_DECLARE_ERROR(27300)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VIDEO_NOT_ENABLED Macro

Syntax

```
#define PRL_ERR_VMCONF_VIDEO_NOT_ENABLED PRL_RESULT_DECLARE_ERROR(27018)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VIRTIO_BLOCK_DEVICES_DUPLICATE_STACK_INDEX Macro

Syntax

```
#define PRL_ERR_VMCONF_VIRTIO_BLOCK_DEVICES_DUPLICATE_STACK_INDEX
PRL_RESULT_DECLARE_ERROR(27704)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VM_NAME_HAS_INVALID_SYMBOL Macro

Syntax

```
#define PRL_ERR_VMCONF_VM_NAME_HAS_INVALID_SYMBOL PRL_RESULT_DECLARE_ERROR(27004)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VM_NAME_IS_EMPTY Macro

Syntax

```
#define PRL_ERR_VMCONF_VM_NAME_IS_EMPTY PRL_RESULT_DECLARE_ERROR(27001)
```

File

PrlErrorsValues.h

PRL_ERR_VMCONF_VM_NAME_IS_TOO_LONG Macro

Syntax

```
#define PRL_ERR_VMCONF_VM_NAME_IS_TOO_LONG PRL_RESULT_DECLARE_ERROR(27033)
```

File

PrlErrorsValues.h

PRL_ERR_VMDIR_INVALID_PATH Macro

Syntax

```
#define PRL_ERR_VMDIR_INVALID_PATH PRL_RESULT_DECLARE_ERROR(285)
```

File

PrlErrorsValues.h

PRL_ERR_VMDIR_PATH_IS_NOT_ABSOLUTE Macro

Syntax

```
#define PRL_ERR_VMDIR_PATH_IS_NOT_ABSOLUTE PRL_RESULT_DECLARE_ERROR(286)
```

File

PrlErrorsValues.h

PRL_ERR_VMHELPER_CREATEVM_FAILED Macro

STW7 VmHelper errors +++ ALIGNED VALUE TO 41400

Syntax

```
#define PRL_ERR_VMHELPER_CREATEVM_FAILED PRL_RESULT_DECLARE_ERROR(41400)
```

File

PrlErrorsValues.h

PRL_ERR_VMHELPER_DIR_PATH_INVALID Macro

Syntax

```
#define PRL_ERR_VMHELPER_DIR_PATH_INVALID PRL_RESULT_DECLARE_ERROR(41404)
```

File

PrlErrorsValues.h

PRL_ERR_VMHELPER_DISK_MOUNT_FAILED Macro

Syntax

```
#define PRL_ERR_VMHELPER_DISK_MOUNT_FAILED PRL_RESULT_DECLARE_ERROR(41402)
```

File

PrlErrorsValues.h

PRL_ERR_VMHELPER_DISK_UNMOUNT_FAILED Macro

Syntax

```
#define PRL_ERR_VMHELPER_DISK_UNMOUNT_FAILED PRL_RESULT_DECLARE_ERROR(41403)
```

File

PrlErrorsValues.h

PRL_ERR_VMHELPER_GETTING_AVAILABLE_DRIVE_FAILED Macro

Syntax

```
#define PRL_ERR_VMHELPER_GETTING_AVAILABLE_DRIVE_FAILED PRL_RESULT_DECLARE_ERROR(41401)
```

File

PrlErrorsValues.h

PRL_ERR_VNC_SERVER_ALREADY_STARTED Macro

VNC server operations errors +++ ALIGNED VALUE TO 35000

Syntax

```
#define PRL_ERR_VNC_SERVER_ALREADY_STARTED PRL_RESULT_DECLARE_ERROR(35000)
```

File

PrlErrorsValues.h

PRL_ERR_VNC_SERVER_AUTOSET_PORT_FAILED Macro

Syntax

```
#define PRL_ERR_VNC_SERVER_AUTOSET_PORT_FAILED PRL_RESULT_DECLARE_ERROR(35002)
```

File

PrlErrorsValues.h

PRL_ERR_VNC_SERVER_DISABLED Macro

Syntax

```
#define PRL_ERR_VNC_SERVER_DISABLED PRL_RESULT_DECLARE_ERROR(35001)
```

File

PrlErrorsValues.h

PRL_ERR_VNC_SERVER_NOT_STARTED Macro

Syntax

```
#define PRL_ERR_VNC_SERVER_NOT_STARTED PRL_RESULT_DECLARE_ERROR(35005)
```

File

PrlErrorsValues.h

PRL_ERR_VOLUME_LICENSE_EXCEEDED_LIMIT Macro

Syntax

```
#define PRL_ERR_VOLUME_LICENSE_EXCEEDED_LIMIT PRL_RESULT_DECLARE_ERROR(11062)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_ALREADY_HOOKED_FAILED Macro

Syntax

```
#define PRL_ERR_VTD_ALREADY_HOOKED_FAILED PRL_RESULT_DECLARE_ERROR(10003)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_DEVICE_DOES_NOT_EXIST Macro

Syntax

```
#define PRL_ERR_VTD_DEVICE_DOES_NOT_EXIST PRL_RESULT_DECLARE_ERROR(460)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_DEVICE_NOT_MAPPED Macro

- VT-d

+++ ALIGNED VALUE TO 12000

Syntax

```
#define PRL_ERR_VTD_DEVICE_NOT_MAPPED PRL_RESULT_DECLARE_ERROR(12000)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_DEVICE_TROUBLESHOOT Macro

Syntax

```
#define PRL_ERR_VTD_DEVICE_TROUBLESHOOT PRL_RESULT_DECLARE_ERROR(10017)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_AFTER_INSTALL_NEED_REBOOT Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_AFTER_INSTALL_NEED_REBOOT PRL_RESULT_DECLARE_ERROR(10007)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_AFTER_REVERT_NEED_REBOOT Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_AFTER_REVERT_NEED_REBOOT PRL_RESULT_DECLARE_ERROR(10008)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_DEVICE_CURRENTLY_IN_USE Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_DEVICE_CURRENTLY_IN_USE PRL_RESULT_DECLARE_ERROR(10012)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_FAILED Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_FAILED PRL_RESULT_DECLARE_ERROR(10004)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_INSTALLATION_FAILED Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_INSTALLATION_FAILED PRL_RESULT_DECLARE_ERROR(10009)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_INVALID_CONFIG Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_INVALID_CONFIG PRL_RESULT_DECLARE_ERROR(10011)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_NEED_REBOOT_FAILED Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_NEED_REBOOT_FAILED PRL_RESULT_DECLARE_ERROR(10005)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_REVERT_FAILED Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_REVERT_FAILED PRL_RESULT_DECLARE_ERROR(10010)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_HOOK_UPDATE_SCRIPT_EXECUTE Macro

Syntax

```
#define PRL_ERR_VTD_HOOK_UPDATE_SCRIPT_EXECUTE PRL_RESULT_DECLARE_ERROR(10019)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_INITIALIZATION_FAILED Macro

Syntax

```
#define PRL_ERR_VTD_INITIALIZATION_FAILED PRL_RESULT_DECLARE_ERROR(10002)
```

File

PrlErrorsValues.h

PRL_ERR_VTD_WAIT_ASR Macro

Syntax

```
#define PRL_ERR_VTD_WAIT_ASR PRL_RESULT_DECLARE_ERROR(10006)
```

File

PrlErrorsValues.h

PRL_ERR_VTX_ENABLED_ONLY_IN_SMX Macro

Syntax

```
#define PRL_ERR_VTX_ENABLED_ONLY_IN_SMX PRL_RESULT_DECLARE_ERROR(342)
```

File

PrlErrorsValues.h

PRL_ERR_VZ_API_NOT_INITIALIZED Macro

Syntax

```
#define PRL_ERR_VZ_API_NOT_INITIALIZED PRL_RESULT_DECLARE_ERROR(27026)
```

File

PrlErrorsValues.h

PRL_ERR_VZ_OPERATION_FAILED Macro

Syntax

```
#define PRL_ERR_VZ_OPERATION_FAILED PRL_RESULT_DECLARE_ERROR(44006)
```

File

PrlErrorsValues.h

PRL_ERR_VZ_OSTEMPLATE_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_VZ_OSTEMPLATE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(44007)
```

File

PrlErrorsValues.h

PRL_ERR_VZCTL_OPERATION_FAILED Macro

Syntax

```
#define PRL_ERR_VZCTL_OPERATION_FAILED PRL_RESULT_DECLARE_ERROR(44010)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_ACTIVATION_KEY Macro

Syntax

```
#define PRL_ERR_VZLICENSE_ACTIVATION_KEY PRL_RESULT_DECLARE_ERROR(11037)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_CANCEL Macro

Syntax

```
#define PRL_ERR_VZLICENSE_CANCEL PRL_RESULT_DECLARE_ERROR(11019)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_EACCES Macro

Syntax

```
#define PRL_ERR_VZLICENSE_EACCES PRL_RESULT_DECLARE_ERROR(11028)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_ERR_KA Macro

Syntax

```
#define PRL_ERR_VZLICENSE_ERR_KA PRL_RESULT_DECLARE_ERROR(11026)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_EXIST Macro

Syntax

```
#define PRL_ERR_VZLICENSE_EXIST PRL_RESULT_DECLARE_ERROR(11031)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_FATAL Macro

Syntax

```
#define PRL_ERR_VZLICENSE_FATAL PRL_RESULT_DECLARE_ERROR(11035)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_INVAL Macro

Syntax

```
#define PRL_ERR_VZLICENSE_INVAL PRL_RESULT_DECLARE_ERROR(11033)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_INVALID_HWID Macro

Syntax

```
#define PRL_ERR_VZLICENSE_INVALID_HWID PRL_RESULT_DECLARE_ERROR(11036)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_IO Macro

Syntax

```
#define PRL_ERR_VZLICENSE_IO PRL_RESULT_DECLARE_ERROR(11032)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_KA_ACTIVATION_LIMIT_REACHED Macro

Syntax

```
#define PRL_ERR_VZLICENSE_KA_ACTIVATION_LIMIT_REACHED PRL_RESULT_DECLARE_ERROR(11066)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_KA_HWID_DOES_NOT_MATCH Macro

Syntax

```
#define PRL_ERR_VZLICENSE_KA_HWID_DOES_NOT_MATCH PRL_RESULT_DECLARE_ERROR(11067)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_KA_LICENSE_EXPIRED Macro

Syntax

```
#define PRL_ERR_VZLICENSE_KA_LICENSE_EXPIRED PRL_RESULT_DECLARE_ERROR(11065)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_KA_LICENSE_IS_NOT_PROLONGATED_YET Macro

Syntax

```
#define PRL_ERR_VZLICENSE_KA_LICENSE_IS_NOT_PROLONGATED_YET PRL_RESULT_DECLARE_ERROR(11064)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_KA_LICENSE_IS_TERMINATED Macro

Syntax

```
#define PRL_ERR_VZLICENSE_KA_LICENSE_IS_TERMINATED PRL_RESULT_DECLARE_ERROR(11068)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_KA_LICENSE_IS_UP_TO_DATE Macro

Syntax

```
#define PRL_ERR_VZLICENSE_KA_LICENSE_IS_UP_TO_DATE PRL_RESULT_DECLARE_ERROR(11063)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_LOCK Macro

Syntax

```
#define PRL_ERR_VZLICENSE_LOCK PRL_RESULT_DECLARE_ERROR(11027)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_NETWORK Macro

Syntax

```
#define PRL_ERR_VZLICENSE_NETWORK PRL_RESULT_DECLARE_ERROR(11022)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_NODATA Macro

Syntax

```
#define PRL_ERR_VZLICENSE_NODATA PRL_RESULT_DECLARE_ERROR(11025)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_NOENT Macro

Syntax

```
#define PRL_ERR_VZLICENSE_NOENT PRL_RESULT_DECLARE_ERROR(11029)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_NOINIT Macro

Syntax

```
#define PRL_ERR_VZLICENSE_NOINIT PRL_RESULT_DECLARE_ERROR(11030)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_NOMEM Macro

Syntax

```
#define PRL_ERR_VZLICENSE_NOMEM PRL_RESULT_DECLARE_ERROR(11034)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_NOTSUP Macro

Syntax

```
#define PRL_ERR_VZLICENSE_NOTSUP PRL_RESULT_DECLARE_ERROR(11023)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_PROXY Macro

Syntax

```
#define PRL_ERR_VZLICENSE_PROXY PRL_RESULT_DECLARE_ERROR(11021)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_PROXY_AUTH Macro

Syntax

```
#define PRL_ERR_VZLICENSE_PROXY_AUTH PRL_RESULT_DECLARE_ERROR(11020)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_TASK_ALREADY_RUN Macro

Syntax

```
#define PRL_ERR_VZLICENSE_TASK_ALREADY_RUN PRL_RESULT_DECLARE_ERROR(11040)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_TIMEOUT Macro

Syntax

```
#define PRL_ERR_VZLICENSE_TIMEOUT PRL_RESULT_DECLARE_ERROR(11018)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_UNSUPPORTED_APP_MODE Macro

Syntax

```
#define PRL_ERR_VZLICENSE_UNSUPPORTED_APP_MODE PRL_RESULT_DECLARE_ERROR(11039)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_UNSUPPORTED_MODE Macro

Syntax

```
#define PRL_ERR_VZLICENSE_UNSUPPORTED_MODE PRL_ERR_VZLICENSE_UNSUPPORTED_APP_MODE
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_UNSUPPORTED_UPDATE_OP Macro

Syntax

```
#define PRL_ERR_VZLICENSE_UNSUPPORTED_UPDATE_OP PRL_RESULT_DECLARE_ERROR(11072)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_VMS_LIMIT_EXCEEDED Macro

Syntax

```
#define PRL_ERR_VZLICENSE_VMS_LIMIT_EXCEEDED PRL_RESULT_DECLARE_ERROR(11038)
```

File

PrlErrorsValues.h

PRL_ERR_VZLICENSE_WRONG Macro

Syntax

```
#define PRL_ERR_VZLICENSE_WRONG PRL_RESULT_DECLARE_ERROR(11024)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_ACCEPTED Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_ACCEPTED PRL_RESULT_DECLARE_SUCCESS(47001)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_ACCOUNT_ALREADY_EXISTS Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_ACCOUNT_ALREADY_EXISTS PRL_ERR_WEB_PORTAL_CONFLICT
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_BAD_REQUEST Macro

errors

Syntax

```
#define PRL_ERR_WEB_PORTAL_BAD_REQUEST PRL_RESULT_DECLARE_ERROR(47002)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_CONFLICT Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_CONFLICT PRL_RESULT_DECLARE_ERROR(47006)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_CREATED Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_CREATED PRL_RESULT_DECLARE_SUCCESS(47000)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_FORBIDDEN Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_FORBIDDEN PRL_RESULT_DECLARE_ERROR(47010)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_INVALID_EMAIL Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_INVALID_EMAIL PRL_RESULT_DECLARE_ERROR(47008)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_INVALID_PARAMETERS Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_INVALID_PARAMETERS PRL_RESULT_DECLARE_ERROR(47005)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_INVALID_PASSWORD Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_INVALID_PASSWORD PRL_ERR_WEB_PORTAL_BAD_REQUEST
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_NO_CONTENT Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_NO_CONTENT PRL_RESULT_DECLARE_SUCCESS(47009)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_NOT_FOUND Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_NOT_FOUND PRL_RESULT_DECLARE_ERROR(47003)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_OK Macro

success

Syntax

```
#define PRL_ERR_WEB_PORTAL_OK PRL_ERR_SUCCESS
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SERVICE_UNAVAILABLE Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SERVICE_UNAVAILABLE PRL_RESULT_DECLARE_ERROR(47007)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SOCIAL_APP_DENIED Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SOCIAL_APP_DENIED PRL_RESULT_DECLARE_ERROR(47013)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SOCIAL_CONFIRM_EMAIL_SENT Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SOCIAL_CONFIRM_EMAIL_SENT PRL_RESULT_DECLARE_ERROR(47011)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SOCIAL_PROHIBITED_FOR_BA Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SOCIAL_PROHIBITED_FOR_BA PRL_RESULT_DECLARE_ERROR(47016)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_BAD_RESPONSE Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_BAD_RESPONSE PRL_RESULT_DECLARE_ERROR(47014)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_UNAVAILABLE Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_UNAVAILABLE PRL_RESULT_DECLARE_ERROR(47015)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_SOCIAL_UNCONFIRMED_ACCOUNT Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_SOCIAL_UNCONFIRMED_ACCOUNT PRL_RESULT_DECLARE_ERROR(47012)
```

File

PrlErrorsValues.h

PRL_ERR_WEB_PORTAL_UNAUTHORIZED Macro

Syntax

```
#define PRL_ERR_WEB_PORTAL_UNAUTHORIZED PRL_RESULT_DECLARE_ERROR(47004)
```

File

PrlErrorsValues.h

PRL_ERR_WIN_AERO_DISABLED Macro

isn't used now

Syntax

```
#define PRL_ERR_WIN_AERO_DISABLED PRL_RESULT_DECLARE_ERROR(10016)
```

File

PrlErrorsValues.h

PRL_ERR_WINDOWS_EXPRESS_INSTALL_USER_NAME_EMPTY Macro

Syntax

```
#define PRL_ERR_WINDOWS_EXPRESS_INSTALL_USER_NAME_EMPTY PRL_RESULT_DECLARE_ERROR(318)
```

File

PrlErrorsValues.h

PRL_ERR_WRITE_FAILED Macro

Syntax

```
#define PRL_ERR_WRITE_FAILED PRL_RESULT_DECLARE_ERROR(519)
```

File

PrlErrorsValues.h

PRL_ERR_WRONG_CONNECTION_SECURITY_LEVEL Macro

Syntax

```
#define PRL_ERR_WRONG_CONNECTION_SECURITY_LEVEL PRL_RESULT_DECLARE_ERROR(277)
```

File

PrlErrorsValues.h

PRL_ERR_WRONG_PROTOCOL_VERSION Macro

Syntax

```
#define PRL_ERR_WRONG_PROTOCOL_VERSION PRL_RESULT_DECLARE_ERROR(413)
```

File

PrlErrorsValues.h

PRL_ERR_WRONG_VM_STATE Macro

Syntax

```
#define PRL_ERR_WRONG_VM_STATE PRL_ERR_COMPACT_WRONG_VM_STATE
```

File

PrlErrorsValues.h

PRL_ERR_WS_DISP_CONNECTION_CLOSED Macro

Syntax

```
#define PRL_ERR_WS_DISP_CONNECTION_CLOSED PRL_RESULT_DECLARE_ERROR(246)
```

File

PrlErrorsValues.h

PRL_ERR_X64GUEST_ON_X32HOST Macro

Syntax

```
#define PRL_ERR_X64GUEST_ON_X32HOST PRL_RESULT_DECLARE_ERROR(354)
```

File

PrlErrorsValues.h

PRL_ERR_X64GUEST_ON_X64VM_HVT_DISABLED Macro

Syntax

```
#define PRL_ERR_X64GUEST_ON_X64VM_HVT_DISABLED PRL_RESULT_DECLARE_ERROR(357)
```

File

PrlErrorsValues.h

PRL_ERR_XML_WRITE_FILE Macro

Syntax

```
#define PRL_ERR_XML_WRITE_FILE PRL_RESULT_DECLARE_ERROR(424)
```

File

PrlErrorsValues.h

PRL_ERR_XMLRPC_INTERNAL_ERROR Macro

Syntax

```
#define PRL_ERR_XMLRPC_INTERNAL_ERROR PRL_RESULT_DECLARE_ERROR(38012)
```

File

PrlErrorsValues.h

PRL_ERR_XMLRPC_INVALID_CREDENTIALS Macro

Syntax

```
#define PRL_ERR_XMLRPC_INVALID_CREDENTIALS PRL_RESULT_DECLARE_ERROR(38009)
```

File

PrlErrorsValues.h

PRL_ERR_XMLRPC_INVALID_REQUEST_INFO Macro

Syntax

```
#define PRL_ERR_XMLRPC_INVALID_REQUEST_INFO PRL_RESULT_DECLARE_ERROR(38011)
```

File

PrlErrorsValues.h

PRL_ERR_XMLRPC_LIMITS_EXHAUSTED Macro

Syntax

```
#define PRL_ERR_XMLRPC_LIMITS_EXHAUSTED PRL_RESULT_DECLARE_ERROR(38010)
```

File

PrlErrorsValues.h

PRL_ERR_XMLRPC_WRONG_METHOD_CALL Macro

Syntax

```
#define PRL_ERR_XMLRPC_WRONG_METHOD_CALL PRL_RESULT_DECLARE_ERROR(38008)
```

File

PrlErrorsValues.h

PRL_GUEST Macros

Macros

Name	Description
PRL_GUEST_CPU_TIME_PTRN (p. 1500)	The sum of guest time changes since the last query (delta) for every vCPU averaged on host CPU number, in microseconds
PRL_GUEST_CPU_USAGE_PTRN (p. 1500)	Guest OS CPU usage in percents (modulated to 100%)
PRL_GUEST_FS_DISK_INDEX_PTRN (p. 1500)	Disk indices (see PrlVmDev_GetIndex (p. 760)() API function)
PRL_GUEST_FS_FREE_SPACE_PTRN (p. 1500)	Amount of free space on a filesystem (KiB)
PRL_GUEST_FS_NAME_PTRN (p. 1501)	Mounted filesystems statistics Filesystem device name, as it is seen inside guest
PRL_GUEST_FS_TOTAL_SPACE_PTRN (p. 1501)	Total size of a filesystem (KiB)
PRL_GUEST_RAM_BALLOON_ACTUAL (p. 1501)	Guest OS balloon sizes in megabytes
PRL_GUEST_RAM_CACHED_PTRN (p. 1501)	
PRL_GUEST_RAM_MAJOR_FAULT (p. 1502)	
PRL_GUEST_RAM_MINOR_FAULT (p. 1502)	
PRL_GUEST_RAM_SWAP_IN (p. 1502)	Guest OS virtual memory stats in counts
PRL_GUEST_RAM_SWAP_OUT (p. 1502)	
PRL_GUEST_RAM_TOTAL_PTRN (p. 1502)	
PRL_GUEST_RAM_USAGE_PTRN (p. 1503)	Guest OS RAM usage in megabytes
PRL_GUEST_VCPU_TIME_PTRN (p. 1503)	per vcpu statistics in nanosecs

PRL_GUEST_CPU_TIME_PTRN Macro

The sum of guest time changes since the last query (delta) for every vCPU averaged on host CPU number, in microseconds

Syntax

```
#define PRL_GUEST_CPU_TIME_PTRN "guest.cpu.time"
```

File

PrlPerfCounters.h

PRL_GUEST_CPU_USAGE_PTRN Macro

Guest OS CPU usage in percents (modulated to 100%)

Syntax

```
#define PRL_GUEST_CPU_USAGE_PTRN "guest.cpu.usage"
```

File

PrlPerfCounters.h

PRL_GUEST_FS_DISK_INDEX_PTRN Macro

Disk indices (see PrlVmDev_GetIndex (p. 760)() API function)

Syntax

```
#define PRL_GUEST_FS_DISK_INDEX_PTRN "guest.fs#.disk.#"
```

File

PrlPerfCounters.h

PRL_GUEST_FS_FREE_SPACE_PTRN Macro

Amount of free space on a filesystem (KiB)

Syntax

```
#define PRL_GUEST_FS_FREE_SPACE_PTRN "guest.fs#.free"
```

File

PrlPerfCounters.h

PRL_GUEST_FS_NAME_PTRN Macro

Mounted filesystems statistics Filesystem device name, as it is seen inside guest

Syntax

```
#define PRL_GUEST_FS_NAME_PTRN "guest.fs#.name"
```

File

PrlPerfCounters.h

PRL_GUEST_FS_TOTAL_SPACE_PTRN Macro

Total size of a filesystem (KiB)

Syntax

```
#define PRL_GUEST_FS_TOTAL_SPACE_PTRN "guest.fs#.total"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_BALLOON_ACTUAL Macro

Guest OS balloon sizes in megabytes

Syntax

```
#define PRL_GUEST_RAM_BALLOON_ACTUAL "guest.ram.balloon_actual"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_CACHED_PTRN Macro

Syntax

```
#define PRL_GUEST_RAM_CACHED_PTRN "guest.ram.cached"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_MAJOR_FAULT Macro

Syntax

```
#define PRL_GUEST_RAM_MAJOR_FAULT "guest.ram.major_fault"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_MINOR_FAULT Macro

Syntax

```
#define PRL_GUEST_RAM_MINOR_FAULT "guest.ram.minor_fault"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_SWAP_IN Macro

Guest OS virtual memory stats in counts

Syntax

```
#define PRL_GUEST_RAM_SWAP_IN "guest.ram.swap_in"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_SWAP_OUT Macro

Syntax

```
#define PRL_GUEST_RAM_SWAP_OUT "guest.ram.swap_out"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_TOTAL_PTRN Macro

Syntax

```
#define PRL_GUEST_RAM_TOTAL_PTRN "guest.ram.total"
```

File

PrlPerfCounters.h

PRL_GUEST_RAM_USAGE_PTRN Macro

Guest OS RAM usage in megabytes

Syntax

```
#define PRL_GUEST_RAM_USAGE_PTRN "guest.ram.usage"
```

File

PrlPerfCounters.h

PRL_GUEST_VCPU_TIME_PTRN Macro

per vcpu statistics in nanosecs

Syntax

```
#define PRL_GUEST_VCPU_TIME_PTRN "guest.vcpu#.time"
```

File

PrlPerfCounters.h

PRL_IMAGE Macros

Macros

Name	Description
PRL_IMAGE_BOOTCAMP (p. 1504)	Bootcamp
PRL_IMAGE_BOOTCAMP_FAKE (p. 1504)	FAKE Bootcamp, for test purposes
PRL_IMAGE_BOOTCAMP_UID (p. 1504)	Bootcamp set by UID
PRL_IMAGE_COMPRESSED (p. 1504)	Compressed disk (version 1)
PRL_IMAGE_INVALID (p. 1505)	Invalid type
PRL_IMAGE_LAST_INVALID (p. 1505)	Last (invalid)
PRL_IMAGE_PARTITION (p. 1505)	Partition from disk
PRL_IMAGE_PHYSICAL (p. 1505)	Real disk (full)
PRL_IMAGE_PLAIN (p. 1506)	Plain disk
PRL_IMAGE_TRY_GUESS (p. 1506)	Software should guess what kind of disk it is
PRL_IMAGE_VHD_DIFF (p. 1506)	
PRL_IMAGE_VHD_DYNAMIC (p. 1506)	
PRL_IMAGE_VHD_DYNAMIC_PART (p. 1507)	
PRL_IMAGE_VHD_PLAIN (p. 1507)	Limited Microsoft VHD support

PRL_IMAGE_VHD_PLAIN_PART (p. 1507)	
PRL_IMAGE_VMDK_EXTENT_2GB_FLAT (p. 1507)	
PRL_IMAGE_VMDK_EXTENT_2GB_SPARSE (p. 1507)	
PRL_IMAGE_VMDK_MONOLITHIC_FLAT (p. 1508)	VMware VMDK support
PRL_IMAGE_VMDK_MONOLITHIC_SPARSE (p. 1508)	

PRL_IMAGE_BOOTCAMP Macro

Bootcamp

Syntax

```
#define PRL_IMAGE_BOOTCAMP 6
```

File

PrlDisk.h

PRL_IMAGE_BOOTCAMP_FAKE Macro

FAKE Bootcamp, for test purposes

Syntax

```
#define PRL_IMAGE_BOOTCAMP_FAKE 8
```

File

PrlDisk.h

PRL_IMAGE_BOOTCAMP_UID Macro

Bootcamp set by UID

Syntax

```
#define PRL_IMAGE_BOOTCAMP_UID 7
```

File

PrlDisk.h

PRL_IMAGE_COMPRESSED Macro

Compressed disk (version 1)

Syntax

```
#define PRL_IMAGE_COMPRESSED 2
```

File

PrlDisk.h

PRL_IMAGE_INVALID Macro

Invalid type

Syntax

```
#define PRL_IMAGE_INVALID 0
```

File

PrlDisk.h

PRL_IMAGE_LAST_INVALID Macro

Last (invalid)

Syntax

```
#define PRL_IMAGE_LAST_INVALID 0x80000000
```

File

PrlDisk.h

PRL_IMAGE_PARTITION Macro

Partition from disk

Syntax

```
#define PRL_IMAGE_PARTITION 5
```

File

PrlDisk.h

PRL_IMAGE_PHYSICAL Macro

Real disk (full)

Syntax

```
#define PRL_IMAGE_PHYSICAL 4
```

File

PrlDisk.h

PRL_IMAGE_PLAIN Macro

Plain disk

Syntax

```
#define PRL_IMAGE_PLAIN 1
```

File

PrlDisk.h

PRL_IMAGE_TRY_GUESS Macro

Software should guess what kind of disk it is

Syntax

```
#define PRL_IMAGE_TRY_GUESS 100
```

File

PrlDisk.h

PRL_IMAGE_VHD_DIFF Macro

Syntax

```
#define PRL_IMAGE_VHD_DIFF 84
```

File

PrlDisk.h

PRL_IMAGE_VHD_DYNAMIC Macro

Syntax

```
#define PRL_IMAGE_VHD_DYNAMIC 81
```

File

PrlDisk.h

PRL_IMAGE_VHD_DYNAMIC_PART Macro

Syntax

```
#define PRL_IMAGE_VHD_DYNAMIC_PART 83
```

File

PrlDisk.h

PRL_IMAGE_VHD_PLAIN Macro

Limited Microsoft VHD support

Syntax

```
#define PRL_IMAGE_VHD_PLAIN 80
```

File

PrlDisk.h

PRL_IMAGE_VHD_PLAIN_PART Macro

Syntax

```
#define PRL_IMAGE_VHD_PLAIN_PART 82
```

File

PrlDisk.h

PRL_IMAGE_VMDK_EXTENT_2GB_FLAT Macro

Syntax

```
#define PRL_IMAGE_VMDK_EXTENT_2GB_FLAT 92
```

File

PrlDisk.h

PRL_IMAGE_VMDK_EXTENT_2GB_SPARSE Macro

Syntax

```
#define PRL_IMAGE_VMDK_EXTENT_2GB_SPARSE 93
```

File

PrlDisk.h

PRL_IMAGE_VMDK_MONOLITHIC_FLAT Macro

VMware VMDK support

Syntax

```
#define PRL_IMAGE_VMDK_MONOLITHIC_FLAT 90
```

File

PrlDisk.h

PRL_IMAGE_VMDK_MONOLITHIC_SPARSE Macro

Syntax

```
#define PRL_IMAGE_VMDK_MONOLITHIC_SPARSE 91
```

File

PrlDisk.h

PRL_NET Macros

Macros

Name	Description
PRL_NET_ADAPTER_ALREADY_USED (p. 1510)	Unable to use adapter because it is already used by another virtual network
PRL_NET_ADAPTER_NOT_EXIST (p. 1510)	
PRL_NET_BIND_FAILED (p. 1510)	
PRL_NET_BYTES_IN_PTRN (p. 1510)	
PRL_NET_BYTES_OUT_PTRN (p. 1511)	
PRL_NET_CABLE_DISCONNECTED (p. 1511)	
PRL_NET_CLASSFUL_TRAFFIC_IPV4_PTRN (p. 1511)	
PRL_NET_CLASSFUL_TRAFFIC_IPV6_PTRN (p. 1511)	
PRL_NET_CLASSFUL_TRAFFIC_PTRN (p. 1511)	Classful network traffic counting info.
PRL_NET_CONNECTION_SHARING_CONFLICT (p. 1512)	
PRL_NET_DUPLICATE_IPPRIVATE_NETWORK_NAME (p. 1512)	

PRL_NET_DUPLICATE_VIRTUAL_NETWORK_ID (p. 1512)	
PRL_NET_ERR_ADAPTER_CONFIG (p. 1512)	
PRL_NET_ERR_ETH_NO_BINDABLE_ADAPTER (p. 1513)	
PRL_NET_ERR_PRL_NO_BINDABLE_ADAPTER (p. 1513)	
PRL_NET_ETHLIST_CREATE_ERROR (p. 1513)	• Network +++ ALIGNED VALUE TO 4000
PRL_NET_INSTALL_FAILED (p. 1513)	
PRL_NET_INSTALL_TIMEOUT (p. 1513)	
PRL_NET_IPADDRESS_MODIFY_FAILED_WARNING (p. 1514)	Operation succeeded, but failed to change Network Adapter IP-address.
PRL_NET_IPPRIVATE_NETWORK_DOES_NOT_EXIST (p. 1514)	
PRL_NET_IPPRIVATE_NETWORK_INVALID_IP (p. 1514)	
PRL_NET_OFFMGMT_ERROR (p. 1514)	Failed to setup offline-management configuration for VM
PRL_NET_PKTFILTER_ERROR (p. 1515)	Failed to setup packet filter for VM Adapter.
PRL_NET_PKTS_IN_PTRN (p. 1515)	
PRL_NET_PKTS_OUT_PTRN (p. 1515)	
PRL_NET_PRLNET_OPEN_FAILED (p. 1515)	
PRL_NET_RENAME_FAILED (p. 1516)	
PRL_NET_RESTORE_DEFAULTS_PARTITIAL_SUCCESS (p. 1516)	
PRL_NET_SRV_NOTIFY_ERROR (p. 1516)	
PRL_NET_SYSTEM_ERROR (p. 1516)	
PRL_NET_UNINSTALL_FAILED (p. 1516)	
PRL_NET_VALID_DHCP_RANGE_WRONG_IP_ADDRS (p. 1517)	Invalid DHCP scope ip addresses format
PRL_NET_VALID_DHCP_SCOPE_RANGE_LESS_MIN (p. 1517)	Minimum ip addresses in scope range are 16
PRL_NET_VALID_FAILURE (p. 1517)	+++ ALIGNED VALUE TO 4200 +++ Broken network config
PRL_NET_VALID_MISMATCH_DHCP_SCOPE_MASK (p. 1517)	Mask must match DHCP scope ip addresses
PRL_NET_VALID_WRONG_DHCP_IP_ADDR (p. 1518)	Invalid DHCP ip address format
PRL_NET_VALID_WRONG_HOST_IP_ADDR (p. 1518)	Invalid host ip address format
PRL_NET_VALID_WRONG_NET_MASK (p. 1518)	Invalid network mask format
PRL_NET_VIRTUAL_NETWORK_ID_NOT_EXISTS (p. 1518)	
PRL_NET_VIRTUAL_NETWORK_NOT_FOUND (p. 1519)	
PRL_NET_VIRTUAL_NETWORK_SHARED_EXISTS (p. 1519)	Unable to add shared network because it is already exists
PRL_NET_VIRTUAL_NETWORK_SHARED_PROHIBITED (p. 1519)	Shared network prohibited on server
PRL_NET_VLAN_UNSUPPORTED_IN_THIS_VERSION (p. 1519)	

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_CONFIG_ERROR (p. 1520)	Error while connect VMDevice: Misconfiguration in virtual network.
PRL_NET_VMDEVICE_VIRTUAL_NETWORK_DISABLED (p. 1520)	Error while connect VMDevice: Specified virtual network is disabled.
PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NO_ADAPTER (p. 1520)	Error while connect VMDevice: Adapter for virtual network does not exist.
PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NOT_EXIST (p. 1520)	Error while connect VMDevice: network for VM Network Adapter does not exist.
PRL_NET_WINSCM_OPEN_ERROR (p. 1521)	

PRL_NET_ADAPTER_ALREADY_USED Macro

Unable to use adapter because it is already used by another virtual network

Syntax

```
#define PRL_NET_ADAPTER_ALREADY_USED PRL_RESULT_DECLARE_ERROR(4027)
```

File

PrlErrorsValues.h

PRL_NET_ADAPTER_NOT_EXIST Macro

Syntax

```
#define PRL_NET_ADAPTER_NOT_EXIST PRL_RESULT_DECLARE_ERROR(4004)
```

File

PrlErrorsValues.h

PRL_NET_BIND_FAILED Macro

Syntax

```
#define PRL_NET_BIND_FAILED PRL_RESULT_DECLARE_ERROR(4009)
```

File

PrlErrorsValues.h

PRL_NET_BYTES_IN_PTRN Macro

Syntax

```
#define PRL_NET_BYTES_IN_PTRN "net.nic#.bytes_in"
```

File

PrlPerfCounters.h

PRL_NET_BYTES_OUT_PTRN Macro

Syntax

```
#define PRL_NET_BYTES_OUT_PTRN "net.nic#.bytes_out"
```

File

PrlPerfCounters.h

PRL_NET_CABLE_DISCONNECTED Macro

Syntax

```
#define PRL_NET_CABLE_DISCONNECTED PRL_RESULT_DECLARE_ERROR(4010)
```

File

PrlErrorsValues.h

PRL_NET_CLASSFUL_TRAFFIC_IPV4_PTRN Macro

Syntax

```
#define PRL_NET_CLASSFUL_TRAFFIC_IPV4_PTRN "net.classful.traffic.ipv4"
```

File

PrlPerfCounters.h

PRL_NET_CLASSFUL_TRAFFIC_IPV6_PTRN Macro

Syntax

```
#define PRL_NET_CLASSFUL_TRAFFIC_IPV6_PTRN "net.classful.traffic.ipv6"
```

File

PrlPerfCounters.h

PRL_NET_CLASSFUL_TRAFFIC_PTRN Macro

Classful network traffic counting info.

Syntax

```
#define PRL_NET_CLASSFUL_TRAFFIC_PTRN "net.classful.traffic"
```

File

PrlPerfCounters.h

Remarks

Result parameter provides as binary buffer contains structure PRL_STAT_NET_TRAFFIC. See PrlIOStructs.h for more info

PRL_NET_CONNECTION_SHARING_CONFLICT Macro

Syntax

```
#define PRL_NET_CONNECTION_SHARING_CONFLICT PRL_RESULT_DECLARE_ERROR(4015)
```

File

PrlErrorsValues.h

PRL_NET_DUPLICATE_IPPRIVATE_NETWORK_NAME Macro

Syntax

```
#define PRL_NET_DUPLICATE_IPPRIVATE_NETWORK_NAME PRL_RESULT_DECLARE_ERROR(4031)
```

File

PrlErrorsValues.h

PRL_NET_DUPLICATE_VIRTUAL_NETWORK_ID Macro

Syntax

```
#define PRL_NET_DUPLICATE_VIRTUAL_NETWORK_ID PRL_RESULT_DECLARE_ERROR(4024)
```

File

PrlErrorsValues.h

PRL_NET_ERR_ADAPTER_CONFIG Macro

Syntax

```
#define PRL_NET_ERR_ADAPTER_CONFIG PRL_RESULT_DECLARE_ERROR(4005)
```

File

PrlErrorsValues.h

PRL_NET_ERR_ETH_NO_BINDABLE_ADAPTER Macro

Syntax

```
#define PRL_NET_ERR_ETH_NO_BINDABLE_ADAPTER PRL_RESULT_DECLARE_ERROR(4006)
```

File

PrlErrorsValues.h

PRL_NET_ERR_PRL_NO_BINDABLE_ADAPTER Macro

Syntax

```
#define PRL_NET_ERR_PRL_NO_BINDABLE_ADAPTER PRL_RESULT_DECLARE_ERROR(4007)
```

File

PrlErrorsValues.h

PRL_NET_ETHLIST_CREATE_ERROR Macro

- Network

+++ ALIGNED VALUE TO 4000

Syntax

```
#define PRL_NET_ETHLIST_CREATE_ERROR PRL_RESULT_DECLARE_ERROR(4000)
```

File

PrlErrorsValues.h

PRL_NET_INSTALL_FAILED Macro

Syntax

```
#define PRL_NET_INSTALL_FAILED PRL_RESULT_DECLARE_ERROR(4013)
```

File

PrlErrorsValues.h

PRL_NET_INSTALL_TIMEOUT Macro

Syntax

```
#define PRL_NET_INSTALL_TIMEOUT PRL_RESULT_DECLARE_ERROR(4012)
```

File

PrlErrorsValues.h

PRL_NET_IPADDRESS_MODIFY_FAILED_WARNING Macro

Operation succeeded, but failed to change Network Adapter IP-address.

Syntax

```
#define PRL_NET_IPADDRESS_MODIFY_FAILED_WARNING PRL_RESULT_DECLARE_SUCCESS(4022)
```

File

PrlErrorsValues.h

PRL_NET_IPPRIVATE_NETWORK_DOES_NOT_EXIST Macro

Syntax

```
#define PRL_NET_IPPRIVATE_NETWORK_DOES_NOT_EXIST PRL_RESULT_DECLARE_ERROR(4032)
```

File

PrlErrorsValues.h

PRL_NET_IPPRIVATE_NETWORK_INVALID_IP Macro

Syntax

```
#define PRL_NET_IPPRIVATE_NETWORK_INVALID_IP PRL_RESULT_DECLARE_ERROR(4033)
```

File

PrlErrorsValues.h

PRL_NET_OFFMGMT_ERROR Macro

Failed to setup offline-management configuration for VM

Syntax

```
#define PRL_NET_OFFMGMT_ERROR PRL_RESULT_DECLARE_ERROR(4029)
```

File

PrlErrorsValues.h

PRL_NET_PKTFILTER_ERROR Macro

Failed to setup packet filter for VM Adapter.

Syntax

```
#define PRL_NET_PKTFILTER_ERROR PRL_RESULT_DECLARE_ERROR(4023)
```

File

PrlErrorsValues.h

Remarks

Adapter will be disconnected.

PRL_NET_PKTS_IN_PTRN Macro

Syntax

```
#define PRL_NET_PKTS_IN_PTRN "net.nic#.pkts_in"
```

File

PrlPerfCounters.h

PRL_NET_PKTS_OUT_PTRN Macro

Syntax

```
#define PRL_NET_PKTS_OUT_PTRN "net.nic#.pkts_out"
```

File

PrlPerfCounters.h

PRL_NET_PRLNET_OPEN_FAILED Macro

Syntax

```
#define PRL_NET_PRLNET_OPEN_FAILED PRL_RESULT_DECLARE_ERROR(4008)
```

File

PrlErrorsValues.h

PRL_NET_RENAME_FAILED Macro

Syntax

```
#define PRL_NET_RENAME_FAILED PRL_RESULT_DECLARE_ERROR(4011)
```

File

PrlErrorsValues.h

PRL_NET_RESTORE_DEFAULTS_PARTITIAL_SUCCESS Macro

Syntax

```
#define PRL_NET_RESTORE_DEFAULTS_PARTITIAL_SUCCESS PRL_RESULT_DECLARE_ERROR(4016)
```

File

PrlErrorsValues.h

PRL_NET_SRV_NOTIFY_ERROR Macro

Syntax

```
#define PRL_NET_SRV_NOTIFY_ERROR PRL_RESULT_DECLARE_ERROR(4003)
```

File

PrlErrorsValues.h

PRL_NET_SYSTEM_ERROR Macro

Syntax

```
#define PRL_NET_SYSTEM_ERROR PRL_RESULT_DECLARE_ERROR(4001)
```

File

PrlErrorsValues.h

PRL_NET_UNINSTALL_FAILED Macro

Syntax

```
#define PRL_NET_UNINSTALL_FAILED PRL_RESULT_DECLARE_ERROR(4014)
```

File

PrlErrorsValues.h

PRL_NET_VALID_DHCP_RANGE_WRONG_IP_ADDRS Macro

Invalid DHCP scope ip addresses format

Syntax

```
#define PRL_NET_VALID_DHCP_RANGE_WRONG_IP_ADDRS PRL_RESULT_DECLARE_ERROR(4204)
```

File

PrlErrorsValues.h

PRL_NET_VALID_DHCP_SCOPE_RANGE_LESS_MIN Macro

Minimum ip addresses in scope range are 16

Syntax

```
#define PRL_NET_VALID_DHCP_SCOPE_RANGE_LESS_MIN PRL_RESULT_DECLARE_ERROR(4206)
```

File

PrlErrorsValues.h

PRL_NET_VALID_FAILURE Macro

+++ ALIGNED VALUE TO 4200 +++ Broken network config

Syntax

```
#define PRL_NET_VALID_FAILURE PRL_RESULT_DECLARE_ERROR(4200)
```

File

PrlErrorsValues.h

PRL_NET_VALID_MISMATCH_DHCP_SCOPE_MASK Macro

Mask must match DHCP scope ip addresses

Syntax

```
#define PRL_NET_VALID_MISMATCH_DHCP_SCOPE_MASK PRL_RESULT_DECLARE_ERROR(4205)
```

File

PrlErrorsValues.h

PRL_NET_VALID_WRONG_DHCP_IP_ADDR Macro

Invalid DHCP ip address format

Syntax

```
#define PRL_NET_VALID_WRONG_DHCP_IP_ADDR PRL_RESULT_DECLARE_ERROR(4203)
```

File

PrlErrorsValues.h

PRL_NET_VALID_WRONG_HOST_IP_ADDR Macro

Invalid host ip address format

Syntax

```
#define PRL_NET_VALID_WRONG_HOST_IP_ADDR PRL_RESULT_DECLARE_ERROR(4202)
```

File

PrlErrorsValues.h

PRL_NET_VALID_WRONG_NET_MASK Macro

Invalid network mask format

Syntax

```
#define PRL_NET_VALID_WRONG_NET_MASK PRL_RESULT_DECLARE_ERROR(4201)
```

File

PrlErrorsValues.h

PRL_NET_VIRTUAL_NETWORK_ID_NOT_EXISTS Macro

Syntax

```
#define PRL_NET_VIRTUAL_NETWORK_ID_NOT_EXISTS PRL_RESULT_DECLARE_ERROR(4025)
```

File

PrlErrorsValues.h

PRL_NET_VIRTUAL_NETWORK_NOT_FOUND Macro

Syntax

```
#define PRL_NET_VIRTUAL_NETWORK_NOT_FOUND PRL_RESULT_DECLARE_ERROR(4026)
```

File

PrlErrorsValues.h

PRL_NET_VIRTUAL_NETWORK_SHARED_EXISTS Macro

Unable to add shared network because it is already exists

Syntax

```
#define PRL_NET_VIRTUAL_NETWORK_SHARED_EXISTS PRL_RESULT_DECLARE_ERROR(4028)
```

File

PrlErrorsValues.h

PRL_NET_VIRTUAL_NETWORK_SHARED_PROHIBITED Macro

Shared network prohibited on server

Syntax

```
#define PRL_NET_VIRTUAL_NETWORK_SHARED_PROHIBITED PRL_RESULT_DECLARE_ERROR(4030)
```

File

PrlErrorsValues.h

PRL_NET_VLAN_UNSUPPORTED_IN_THIS_VERSION Macro

Syntax

```
#define PRL_NET_VLAN_UNSUPPORTED_IN_THIS_VERSION PRL_RESULT_DECLARE_ERROR(4017)
```

File

PrlErrorsValues.h

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_CONFIG_ERRO R Macro

Error while connect VMDevice: Misconfiguration in virtual network.

Syntax

```
#define PRL_NET_VMDEVICE_VIRTUAL_NETWORK_CONFIG_ERROR PRL_RESULT_DECLARE_ERROR(4020)
```

File

PrlErrorsValues.h

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_DISABLED Macro

Error while connect VMDevice: Specified virtual network is disabled.

Syntax

```
#define PRL_NET_VMDEVICE_VIRTUAL_NETWORK_DISABLED PRL_RESULT_DECLARE_ERROR(4021)
```

File

PrlErrorsValues.h

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NO_ADAPTER Macro

Error while connect VMDevice: Adapter for virtual network does not exist.

Syntax

```
#define PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NO_ADAPTER PRL_RESULT_DECLARE_ERROR(4019)
```

File

PrlErrorsValues.h

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NOT_EXIST Macro

Error while connect VMDevice: network for VM Network Adapter does not exist.

Syntax

```
#define PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NOT_EXIST PRL_RESULT_DECLARE_ERROR(4018)
```

File

PrlErrorsValues.h

PRL_NET_WINSCM_OPEN_ERROR Macro

Syntax

```
#define PRL_NET_WINSCM_OPEN_ERROR PRL_RESULT_DECLARE_ERROR(4002)
```

File

PrlErrorsValues.h

PRL_QUEST Macros

Macros

Name	Description
PRL_QUEST_CD_DRIVE_NOT_EXIST (p. 1521)	
PRL_QUEST_CDD_IMAGE_NOT_EXIST (p. 1522)	
PRL_QUEST_EXCEED_RECOMMENDED_MEMORY (p. 1522)	
PRL_QUEST_FDD_DRIVE_NOT_EXIST (p. 1522)	
PRL_QUEST_FDD_IMAGE_NOT_EXIST (p. 1522)	
PRL_QUEST_HDD_IMAGE_NOT_EXIST (p. 1522)	
PRL_QUEST_HDD_SAME_IMAGE (p. 1523)	
PRL_QUEST_POWER_OFF (p. 1523)	
PRL_QUEST_RESET (p. 1523)	

PRL_QUEST_CD_DRIVE_NOT_EXIST Macro

Syntax

```
#define PRL_QUEST_CD_DRIVE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(221)
```

File

PrlErrorsValues.h

PRL_QUEST_CDD_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_QUEST_CDD_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(220)
```

File

PrlErrorsValues.h

PRL_QUEST_EXCEED_RECOMMENDED_MEMORY Macro

Syntax

```
#define PRL_QUEST_EXCEED_RECOMMENDED_MEMORY PRL_RESULT_DECLARE_ERROR(219)
```

File

PrlErrorsValues.h

PRL_QUEST_FDD_DRIVE_NOT_EXIST Macro

Syntax

```
#define PRL_QUEST_FDD_DRIVE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(225)
```

File

PrlErrorsValues.h

PRL_QUEST_FDD_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_QUEST_FDD_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(224)
```

File

PrlErrorsValues.h

PRL_QUEST_HDD_IMAGE_NOT_EXIST Macro

Syntax

```
#define PRL_QUEST_HDD_IMAGE_NOT_EXIST PRL_RESULT_DECLARE_ERROR(223)
```

File

PrlErrorsValues.h

PRL_QUEST_HDD_SAME_IMAGE Macro

Syntax

```
#define PRL_QUEST_HDD_SAME_IMAGE PRL_RESULT_DECLARE_ERROR(222)
```

File

PrlErrorsValues.h

PRL_QUEST_POWER_OFF Macro

Syntax

```
#define PRL_QUEST_POWER_OFF PRL_RESULT_DECLARE_ERROR(311)
```

File

PrlErrorsValues.h

PRL_QUEST_RESET Macro

Syntax

```
#define PRL_QUEST_RESET PRL_RESULT_DECLARE_ERROR(312)
```

File

PrlErrorsValues.h

PRL_QUESTION Macros

Macros

Name	Description
PRL_QUESTION_ASK_ENCRYPTED_VM_PASSWORD (p. 1524)	
PRL_QUESTION_CAN_NOT_GET_DISK_FREE_SPACE (p. 1524)	
PRL_QUESTION_CONVERT_3RD_PARTY_CANNOT_MIGRATE (p. 1524)	
PRL_QUESTION_CONVERT_GUEST_OS_IS_HIBERNATED (p. 1524)	
PRL_QUESTION_CONVERT_VM_CANT_DETECT_OS (p. 1525)	
PRL_QUESTION_CONVERT_VM_SPECIFY_DISTRO_PATH (p. 1525)	
PRL_QUESTION_FINALIZE_VM_PROCESS (p. 1525)	

PRL_QUESTION_SETUP_YANDEX_SEARCH (p. 1525)	

PRL_QUESTION_ASK_ENCRYPTED_VM_PASSWORD Macro

Syntax

```
#define PRL_QUESTION_ASK_ENCRYPTED_VM_PASSWORD PRL_RESULT_DECLARE_SUCCESS(13043)
```

File

PrlErrorsValues.h

PRL_QUESTION_CAN_NOT_GET_DISK_FREE_SPACE Macro

Syntax

```
#define PRL_QUESTION_CAN_NOT_GET_DISK_FREE_SPACE PRL_ERR_CAN_NOT_GET_DISK_FREE_SPACE
```

File

PrlErrorsValues.h

PRL_QUESTION_CONVERT_3RD_PARTY_CANNOT_MIGRATE Macro

Syntax

```
#define PRL_QUESTION_CONVERT_3RD_PARTY_CANNOT_MIGRATE PRL_RESULT_DECLARE_SUCCESS(13040)
```

File

PrlErrorsValues.h

PRL_QUESTION_CONVERT_GUEST_OS_IS_HIBERNATED Macro

Syntax

```
#define PRL_QUESTION_CONVERT_GUEST_OS_IS_HIBERNATED PRL_RESULT_DECLARE_SUCCESS(13055)
```

File

PrlErrorsValues.h

PRL_QUESTION_CONVERT_VM_CANT_DETECT_OS Macro

Syntax

```
#define PRL_QUESTION_CONVERT_VM_CANT_DETECT_OS PRL_RESULT_DECLARE_SUCCESS(13045)
```

File

PrlErrorsValues.h

PRL_QUESTION_CONVERT_VM_SPECIFY_DISTRO_PATH Macro

Syntax

```
#define PRL_QUESTION_CONVERT_VM_SPECIFY_DISTRO_PATH PRL_RESULT_DECLARE_SUCCESS(13042)
```

File

PrlErrorsValues.h

PRL_QUESTION_FINALIZE_VM_PROCESS Macro

Syntax

```
#define PRL_QUESTION_FINALIZE_VM_PROCESS PRL_RESULT_DECLARE_SUCCESS(13049)
```

File

PrlErrorsValues.h

PRL_QUESTION_SETUP_YANDEX_SEARCH Macro

Syntax

```
#define PRL_QUESTION_SETUP_YANDEX_SEARCH PRL_RESULT_DECLARE_SUCCESS(13056)
```

File

PrlErrorsValues.h

PRL_SCAN_CODE Macros

Macros

Name	Description
PRL_SCAN_CODE_0 (p. 1529)	
PRL_SCAN_CODE_1 (p. 1530)	
PRL_SCAN_CODE_2 (p. 1530)	
PRL_SCAN_CODE_3 (p. 1530)	
PRL_SCAN_CODE_4 (p. 1530)	
PRL_SCAN_CODE_5 (p. 1530)	
PRL_SCAN_CODE_6 (p. 1531)	
PRL_SCAN_CODE_7 (p. 1531)	
PRL_SCAN_CODE_8 (p. 1531)	
PRL_SCAN_CODE_9 (p. 1531)	
PRL_SCAN_CODE_A (p. 1531)	
PRL_SCAN_CODE_ALT_LEFT (p. 1532)	
PRL_SCAN_CODE_ALT_RIGHT (p. 1532)	
PRL_SCAN_CODE_APP_CALCULATOR (p. 1532)	
PRL_SCAN_CODE_APP_MAIL (p. 1532)	
PRL_SCAN_CODE_APP_MY_COMPUTER (p. 1532)	
PRL_SCAN_CODE_B (p. 1533)	
PRL_SCAN_CODE_BACKSLASH (p. 1533)	
PRL_SCAN_CODE_BACKSPACE (p. 1533)	
PRL_SCAN_CODE_BRAZILIAN_KEYPAD (p. 1533)	
PRL_SCAN_CODE_BREAK (p. 1533)	
PRL_SCAN_CODE_C (p. 1534)	
PRL_SCAN_CODE_CAPS_LOCK (p. 1534)	
PRL_SCAN_CODE_CBRACE_LEFT (p. 1534)	
PRL_SCAN_CODE_CBRACE_RIGHT (p. 1534)	
PRL_SCAN_CODE_CMD_LEFT (p. 1534)	
PRL_SCAN_CODE_CMD_RIGHT (p. 1535)	
PRL_SCAN_CODE_COLON (p. 1535)	
PRL_SCAN_CODE_CTRL_LEFT (p. 1535)	
PRL_SCAN_CODE_CTRL_RIGHT (p. 1535)	
PRL_SCAN_CODE_D (p. 1535)	
PRL_SCAN_CODE_DELETE (p. 1536)	
PRL_SCAN_CODE_DOLLAR (p. 1536)	
PRL_SCAN_CODE_DOWN (p. 1536)	
PRL_SCAN_CODE_E (p. 1536)	
PRL_SCAN_CODE_EJECT (p. 1536)	

PRL_SCAN_CODE_END (p. 1537)	
PRL_SCAN_CODE_ENTER (p. 1537)	
PRL_SCAN_CODE_ESC (p. 1537)	PS/2 scan codes symbolical names for the SDK purposes.
PRL_SCAN_CODE_EURO (p. 1537)	
PRL_SCAN_CODE_EUROPE_1 (p. 1538)	
PRL_SCAN_CODE_F (p. 1538)	
PRL_SCAN_CODE_F1 (p. 1538)	
PRL_SCAN_CODE_F10 (p. 1538)	
PRL_SCAN_CODE_F11 (p. 1538)	
PRL_SCAN_CODE_F12 (p. 1539)	
PRL_SCAN_CODE_F13 (p. 1539)	
PRL_SCAN_CODE_F14 (p. 1539)	
PRL_SCAN_CODE_F15 (p. 1539)	
PRL_SCAN_CODE_F16 (p. 1539)	
PRL_SCAN_CODE_F17 (p. 1540)	
PRL_SCAN_CODE_F18 (p. 1540)	
PRL_SCAN_CODE_F19 (p. 1540)	
PRL_SCAN_CODE_F2 (p. 1540)	
PRL_SCAN_CODE_F20 (p. 1540)	
PRL_SCAN_CODE_F21 (p. 1541)	
PRL_SCAN_CODE_F22 (p. 1541)	
PRL_SCAN_CODE_F23 (p. 1541)	
PRL_SCAN_CODE_F24 (p. 1541)	
PRL_SCAN_CODE_F3 (p. 1541)	
PRL_SCAN_CODE_F4 (p. 1542)	
PRL_SCAN_CODE_F5 (p. 1542)	
PRL_SCAN_CODE_F6 (p. 1542)	
PRL_SCAN_CODE_F7 (p. 1542)	
PRL_SCAN_CODE_F8 (p. 1542)	
PRL_SCAN_CODE_F9 (p. 1543)	
PRL_SCAN_CODE_G (p. 1543)	
PRL_SCAN_CODE_GREATER (p. 1543)	
PRL_SCAN_CODE_H (p. 1543)	
PRL_SCAN_CODE_HANGUEL (p. 1543)	
PRL_SCAN_CODE_HANJA (p. 1544)	
PRL_SCAN_CODE_HENKAN (p. 1544)	
PRL_SCAN_CODE_HIRAGANA (p. 1544)	
PRL_SCAN_CODE_HIRAGANA_KATAKANA (p. 1544)	
PRL_SCAN_CODE_HOME (p. 1544)	
PRL_SCAN_CODE_I (p. 1545)	
PRL_SCAN_CODE_INSERT (p. 1545)	
PRL_SCAN_CODE_J (p. 1545)	
PRL_SCAN_CODE_K (p. 1545)	
PRL_SCAN_CODE_KATAKANA (p. 1545)	

PRL_SCAN_CODE_L (p. 1546)	
PRL_SCAN_CODE_LEFT (p. 1546)	
PRL_SCAN_CODE_LESS (p. 1546)	
PRL_SCAN_CODE_LESS_GREATER (p. 1546)	
PRL_SCAN_CODE_M (p. 1546)	
PRL_SCAN_CODE_MEDIA_NEXT_TRACK (p. 1547)	
PRL_SCAN_CODE_MEDIA_PLAY_PAUSE (p. 1547)	
PRL_SCAN_CODE_MEDIA_PREV_TRACK (p. 1547)	
PRL_SCAN_CODE_MEDIA_SELECT (p. 1547)	
PRL_SCAN_CODE_MEDIA_STOP (p. 1547)	
PRL_SCAN_CODE_MENU (p. 1548)	
PRL_SCAN_CODE_MINUS (p. 1548)	
PRL_SCAN_CODE_MUHENKAN (p. 1548)	
PRL_SCAN_CODE_MUTE (p. 1548)	
PRL_SCAN_CODE_N (p. 1548)	
PRL_SCAN_CODE_NUM_LOCK (p. 1549)	
PRL_SCAN_CODE_O (p. 1549)	
PRL_SCAN_CODE_P (p. 1549)	
PRL_SCAN_CODE_PAD_0 (p. 1549)	
PRL_SCAN_CODE_PAD_1 (p. 1549)	
PRL_SCAN_CODE_PAD_2 (p. 1550)	
PRL_SCAN_CODE_PAD_3 (p. 1550)	
PRL_SCAN_CODE_PAD_4 (p. 1550)	
PRL_SCAN_CODE_PAD_5 (p. 1550)	
PRL_SCAN_CODE_PAD_6 (p. 1550)	
PRL_SCAN_CODE_PAD_7 (p. 1551)	
PRL_SCAN_CODE_PAD_8 (p. 1551)	
PRL_SCAN_CODE_PAD_9 (p. 1551)	
PRL_SCAN_CODE_PAD_DEL (p. 1551)	
PRL_SCAN_CODE_PAD_ENTER (p. 1551)	
PRL_SCAN_CODE_PAD_EQUAL (p. 1552)	
PRL_SCAN_CODE_PAD_MINUS (p. 1552)	
PRL_SCAN_CODE_PAD_PLUS (p. 1552)	
PRL_SCAN_CODE_PAD_SLASH (p. 1552)	
PRL_SCAN_CODE_PAD_STAR (p. 1552)	
PRL_SCAN_CODE_PAGE_DOWN (p. 1553)	
PRL_SCAN_CODE_PAGE_UP (p. 1553)	
PRL_SCAN_CODE_PAUSE_BREAK (p. 1553)	
PRL_SCAN_CODE_PC9800_KEYPAD (p. 1553)	
PRL_SCAN_CODE_PLUS (p. 1553)	
PRL_SCAN_CODE_PRINT_SCREEN (p. 1554)	
PRL_SCAN_CODE_PRINT_SCREEN2 (p. 1554)	
PRL_SCAN_CODE_Q (p. 1554)	

PRL_SCAN_CODE_QUOTE (p. 1554)	
PRL_SCAN_CODE_R (p. 1554)	
PRL_SCAN_CODE_RIGHT (p. 1555)	
PRL_SCAN_CODE_RO (p. 1555)	
PRL_SCAN_CODE_S (p. 1555)	
PRL_SCAN_CODE_SCROLL_LOCK (p. 1555)	
PRL_SCAN_CODE_SHIFT_LEFT (p. 1555)	
PRL_SCAN_CODE_SHIFT_RIGHT (p. 1556)	
PRL_SCAN_CODE_SLASH (p. 1556)	
PRL_SCAN_CODE_SPACE (p. 1556)	
PRL_SCAN_CODE_SYSRQ (p. 1556)	
PRL_SCAN_CODE_SYSTEM_POWER (p. 1556)	
PRL_SCAN_CODE_SYSTEM_SLEEP (p. 1557)	
PRL_SCAN_CODE_SYSTEM_WAKE (p. 1557)	
PRL_SCAN_CODE_T (p. 1557)	
PRL_SCAN_CODE_TAB (p. 1557)	
PRL_SCAN_CODE_TILDA (p. 1557)	
PRL_SCAN_CODE_U (p. 1558)	
PRL_SCAN_CODE_UP (p. 1558)	
PRL_SCAN_CODE_V (p. 1558)	
PRL_SCAN_CODE_VOLUME_DOWN (p. 1558)	
PRL_SCAN_CODE_VOLUME_UP (p. 1558)	
PRL_SCAN_CODE_W (p. 1559)	
PRL_SCAN_CODE_WWW_BACK (p. 1559)	
PRL_SCAN_CODE_WWW_FAVORITES (p. 1559)	
PRL_SCAN_CODE_WWW_FORWARD (p. 1559)	
PRL_SCAN_CODE_WWW_HOME (p. 1559)	
PRL_SCAN_CODE_WWW_REFRESH (p. 1560)	
PRL_SCAN_CODE_WWW_SEARCH (p. 1560)	
PRL_SCAN_CODE_WWW_STOP (p. 1560)	
PRL_SCAN_CODE_X (p. 1560)	
PRL_SCAN_CODE_Y (p. 1560)	
PRL_SCAN_CODE_YEN (p. 1561)	
PRL_SCAN_CODE_Z (p. 1561)	
PRL_SCAN_CODE_ZENKAKU_HANKAKU (p. 1561)	

PRL_SCAN_CODE_0 Macro

Syntax

```
#define PRL_SCAN_CODE_0 {11}
```

File

PrlKeys.h

PRL_SCAN_CODE_1 Macro

Syntax

```
#define PRL_SCAN_CODE_1 {2}
```

File

PrlKeys.h

PRL_SCAN_CODE_2 Macro

Syntax

```
#define PRL_SCAN_CODE_2 {3}
```

File

PrlKeys.h

PRL_SCAN_CODE_3 Macro

Syntax

```
#define PRL_SCAN_CODE_3 {4}
```

File

PrlKeys.h

PRL_SCAN_CODE_4 Macro

Syntax

```
#define PRL_SCAN_CODE_4 {5}
```

File

PrlKeys.h

PRL_SCAN_CODE_5 Macro

Syntax

```
#define PRL_SCAN_CODE_5 {6}
```

File

PrlKeys.h

PRL_SCAN_CODE_6 Macro

Syntax

```
#define PRL_SCAN_CODE_6 {7}
```

File

PrlKeys.h

PRL_SCAN_CODE_7 Macro

Syntax

```
#define PRL_SCAN_CODE_7 {8}
```

File

PrlKeys.h

PRL_SCAN_CODE_8 Macro

Syntax

```
#define PRL_SCAN_CODE_8 {9}
```

File

PrlKeys.h

PRL_SCAN_CODE_9 Macro

Syntax

```
#define PRL_SCAN_CODE_9 {10}
```

File

PrlKeys.h

PRL_SCAN_CODE_A Macro

Syntax

```
#define PRL_SCAN_CODE_A {30}
```

File

PrlKeys.h

PRL_SCAN_CODE_ALT_LEFT Macro

Syntax

```
#define PRL_SCAN_CODE_ALT_LEFT {56}
```

File

PrlKeys.h

PRL_SCAN_CODE_ALT_RIGHT Macro

Syntax

```
#define PRL_SCAN_CODE_ALT_RIGHT {224,56}
```

File

PrlKeys.h

PRL_SCAN_CODE_APP_CALCULATOR Macro

Syntax

```
#define PRL_SCAN_CODE_APP_CALCULATOR {224,33}
```

File

PrlKeys.h

PRL_SCAN_CODE_APP_MAIL Macro

Syntax

```
#define PRL_SCAN_CODE_APP_MAIL {224,108}
```

File

PrlKeys.h

PRL_SCAN_CODE_APP_MY_COMPUTER Macro

Syntax

```
#define PRL_SCAN_CODE_APP_MY_COMPUTER {224,107}
```

File

PrlKeys.h

PRL_SCAN_CODE_B Macro

Syntax

```
#define PRL_SCAN_CODE_B {48}
```

File

PrlKeys.h

PRL_SCAN_CODE_BACKSLASH Macro

Syntax

```
#define PRL_SCAN_CODE_BACKSLASH {43}
```

File

PrlKeys.h

PRL_SCAN_CODE_BACKSPACE Macro

Syntax

```
#define PRL_SCAN_CODE_BACKSPACE {14}
```

File

PrlKeys.h

PRL_SCAN_CODE_BRAZILIAN_KEYPAD Macro

Syntax

```
#define PRL_SCAN_CODE_BRAZILIAN_KEYPAD {126}
```

File

PrlKeys.h

PRL_SCAN_CODE_BREAK Macro

Syntax

```
#define PRL_SCAN_CODE_BREAK {224,70,224,198}
```

File

PrlKeys.h

PRL_SCAN_CODE_C Macro

Syntax

```
#define PRL_SCAN_CODE_C {46}
```

File

PrlKeys.h

PRL_SCAN_CODE_CAPS_LOCK Macro

Syntax

```
#define PRL_SCAN_CODE_CAPS_LOCK {58}
```

File

PrlKeys.h

PRL_SCAN_CODE_CBRACE_LEFT Macro

Syntax

```
#define PRL_SCAN_CODE_CBRACE_LEFT {26}
```

File

PrlKeys.h

PRL_SCAN_CODE_CBRACE_RIGHT Macro

Syntax

```
#define PRL_SCAN_CODE_CBRACE_RIGHT {27}
```

File

PrlKeys.h

PRL_SCAN_CODE_CMD_LEFT Macro

Syntax

```
#define PRL_SCAN_CODE_CMD_LEFT {224,91}
```

File

PrlKeys.h

PRL_SCAN_CODE_CMD_RIGHT Macro

Syntax

```
#define PRL_SCAN_CODE_CMD_RIGHT {224,92}
```

File

PrlKeys.h

PRL_SCAN_CODE_COLON Macro

Syntax

```
#define PRL_SCAN_CODE_COLON {39}
```

File

PrlKeys.h

PRL_SCAN_CODE_CTRL_LEFT Macro

Syntax

```
#define PRL_SCAN_CODE_CTRL_LEFT {29}
```

File

PrlKeys.h

PRL_SCAN_CODE_CTRL_RIGHT Macro

Syntax

```
#define PRL_SCAN_CODE_CTRL_RIGHT {224,29}
```

File

PrlKeys.h

PRL_SCAN_CODE_D Macro

Syntax

```
#define PRL_SCAN_CODE_D {32}
```

File

PrlKeys.h

PRL_SCAN_CODE_DELETE Macro

Syntax

```
#define PRL_SCAN_CODE_DELETE {224,83}
```

File

PrlKeys.h

PRL_SCAN_CODE_DOLLAR Macro

Syntax

```
#define PRL_SCAN_CODE_DOLLAR {224,52}
```

File

PrlKeys.h

PRL_SCAN_CODE_DOWN Macro

Syntax

```
#define PRL_SCAN_CODE_DOWN {224,80}
```

File

PrlKeys.h

PRL_SCAN_CODE_E Macro

Syntax

```
#define PRL_SCAN_CODE_E {18}
```

File

PrlKeys.h

PRL_SCAN_CODE_EJECT Macro

Syntax

```
#define PRL_SCAN_CODE_EJECT {224,99}
```

File

PrlKeys.h

PRL_SCAN_CODE_END Macro

Syntax

```
#define PRL_SCAN_CODE_END {224,79}
```

File

PrlKeys.h

PRL_SCAN_CODE_ENTER Macro

Syntax

```
#define PRL_SCAN_CODE_ENTER {28}
```

File

PrlKeys.h

PRL_SCAN_CODE_ESC Macro

PS/2 scan codes symbolical names for the SDK purposes.

Syntax

```
#define PRL_SCAN_CODE_ESC {1}
```

File

PrlKeys.h

Notes

scancode usage is deprecated, use PRL_KEY_* declared below

PRL_SCAN_CODE_EURO Macro

Syntax

```
#define PRL_SCAN_CODE_EURO {224,51}
```

File

PrlKeys.h

PRL_SCAN_CODE_EUROPE_1 Macro

Syntax

```
#define PRL_SCAN_CODE_EUROPE_1 {43}
```

File

PrlKeys.h

PRL_SCAN_CODE_F Macro

Syntax

```
#define PRL_SCAN_CODE_F {33}
```

File

PrlKeys.h

PRL_SCAN_CODE_F1 Macro

Syntax

```
#define PRL_SCAN_CODE_F1 {59}
```

File

PrlKeys.h

PRL_SCAN_CODE_F10 Macro

Syntax

```
#define PRL_SCAN_CODE_F10 {68}
```

File

PrlKeys.h

PRL_SCAN_CODE_F11 Macro

Syntax

```
#define PRL_SCAN_CODE_F11 {87}
```

File

PrlKeys.h

PRL_SCAN_CODE_F12 Macro

Syntax

```
#define PRL_SCAN_CODE_F12 {88}
```

File

PrlKeys.h

PRL_SCAN_CODE_F13 Macro

Syntax

```
#define PRL_SCAN_CODE_F13 {100}
```

File

PrlKeys.h

PRL_SCAN_CODE_F14 Macro

Syntax

```
#define PRL_SCAN_CODE_F14 {101}
```

File

PrlKeys.h

PRL_SCAN_CODE_F15 Macro

Syntax

```
#define PRL_SCAN_CODE_F15 {102}
```

File

PrlKeys.h

PRL_SCAN_CODE_F16 Macro

Syntax

```
#define PRL_SCAN_CODE_F16 {103}
```

File

PrlKeys.h

PRL_SCAN_CODE_F17 Macro

Syntax

```
#define PRL_SCAN_CODE_F17 {104}
```

File

PrlKeys.h

PRL_SCAN_CODE_F18 Macro

Syntax

```
#define PRL_SCAN_CODE_F18 {105}
```

File

PrlKeys.h

PRL_SCAN_CODE_F19 Macro

Syntax

```
#define PRL_SCAN_CODE_F19 {106}
```

File

PrlKeys.h

PRL_SCAN_CODE_F2 Macro

Syntax

```
#define PRL_SCAN_CODE_F2 {60}
```

File

PrlKeys.h

PRL_SCAN_CODE_F20 Macro

Syntax

```
#define PRL_SCAN_CODE_F20 {107}
```

File

PrlKeys.h

PRL_SCAN_CODE_F21 Macro

Syntax

```
#define PRL_SCAN_CODE_F21 {108}
```

File

PrlKeys.h

PRL_SCAN_CODE_F22 Macro

Syntax

```
#define PRL_SCAN_CODE_F22 {109}
```

File

PrlKeys.h

PRL_SCAN_CODE_F23 Macro

Syntax

```
#define PRL_SCAN_CODE_F23 {110}
```

File

PrlKeys.h

PRL_SCAN_CODE_F24 Macro

Syntax

```
#define PRL_SCAN_CODE_F24 {118}
```

File

PrlKeys.h

PRL_SCAN_CODE_F3 Macro

Syntax

```
#define PRL_SCAN_CODE_F3 {61}
```

File

PrlKeys.h

PRL_SCAN_CODE_F4 Macro

Syntax

```
#define PRL_SCAN_CODE_F4 {62}
```

File

PrlKeys.h

PRL_SCAN_CODE_F5 Macro

Syntax

```
#define PRL_SCAN_CODE_F5 {63}
```

File

PrlKeys.h

PRL_SCAN_CODE_F6 Macro

Syntax

```
#define PRL_SCAN_CODE_F6 {64}
```

File

PrlKeys.h

PRL_SCAN_CODE_F7 Macro

Syntax

```
#define PRL_SCAN_CODE_F7 {65}
```

File

PrlKeys.h

PRL_SCAN_CODE_F8 Macro

Syntax

```
#define PRL_SCAN_CODE_F8 {66}
```

File

PrlKeys.h

PRL_SCAN_CODE_F9 Macro

Syntax

```
#define PRL_SCAN_CODE_F9 {67}
```

File

PrlKeys.h

PRL_SCAN_CODE_G Macro

Syntax

```
#define PRL_SCAN_CODE_G {34}
```

File

PrlKeys.h

PRL_SCAN_CODE_GREATER Macro

Syntax

```
#define PRL_SCAN_CODE_GREATER {52}
```

File

PrlKeys.h

PRL_SCAN_CODE_H Macro

Syntax

```
#define PRL_SCAN_CODE_H {35}
```

File

PrlKeys.h

PRL_SCAN_CODE_HANGUEL Macro

Syntax

```
#define PRL_SCAN_CODE_HANGUEL {242}
```

File

PrlKeys.h

PRL_SCAN_CODE_HANJA Macro

Syntax

```
#define PRL_SCAN_CODE_HANJA {241}
```

File

PrlKeys.h

PRL_SCAN_CODE_HENKAN Macro

Syntax

```
#define PRL_SCAN_CODE_HENKAN {121}
```

File

PrlKeys.h

PRL_SCAN_CODE_HIRAGANA Macro

Syntax

```
#define PRL_SCAN_CODE_HIRAGANA {119}
```

File

PrlKeys.h

PRL_SCAN_CODE_HIRAGANA_KATAKANA Macro

Syntax

```
#define PRL_SCAN_CODE_HIRAGANA_KATAKANA {112}
```

File

PrlKeys.h

PRL_SCAN_CODE_HOME Macro

Syntax

```
#define PRL_SCAN_CODE_HOME {224,71}
```

File

PrlKeys.h

PRL_SCAN_CODE_I Macro

Syntax

```
#define PRL_SCAN_CODE_I {23}
```

File

PrlKeys.h

PRL_SCAN_CODE_INSERT Macro

Syntax

```
#define PRL_SCAN_CODE_INSERT {224,82}
```

File

PrlKeys.h

PRL_SCAN_CODE_J Macro

Syntax

```
#define PRL_SCAN_CODE_J {36}
```

File

PrlKeys.h

PRL_SCAN_CODE_K Macro

Syntax

```
#define PRL_SCAN_CODE_K {37}
```

File

PrlKeys.h

PRL_SCAN_CODE_KATAKANA Macro

Syntax

```
#define PRL_SCAN_CODE_KATAKANA {120}
```

File

PrlKeys.h

PRL_SCAN_CODE_L Macro

Syntax

```
#define PRL_SCAN_CODE_L {38}
```

File

PrlKeys.h

PRL_SCAN_CODE_LEFT Macro

Syntax

```
#define PRL_SCAN_CODE_LEFT {224,75}
```

File

PrlKeys.h

PRL_SCAN_CODE_LESS Macro

Syntax

```
#define PRL_SCAN_CODE_LESS {51}
```

File

PrlKeys.h

PRL_SCAN_CODE_LESS_GREATER Macro

Syntax

```
#define PRL_SCAN_CODE_LESS_GREATER {86}
```

File

PrlKeys.h

PRL_SCAN_CODE_M Macro

Syntax

```
#define PRL_SCAN_CODE_M {50}
```

File

PrlKeys.h

PRL_SCAN_CODE_MEDIA_NEXT_TRACK Macro

Syntax

```
#define PRL_SCAN_CODE_MEDIA_NEXT_TRACK {224,25}
```

File

PrlKeys.h

PRL_SCAN_CODE_MEDIA_PLAY_PAUSE Macro

Syntax

```
#define PRL_SCAN_CODE_MEDIA_PLAY_PAUSE {224,34}
```

File

PrlKeys.h

PRL_SCAN_CODE_MEDIA_PREV_TRACK Macro

Syntax

```
#define PRL_SCAN_CODE_MEDIA_PREV_TRACK {224,16}
```

File

PrlKeys.h

PRL_SCAN_CODE_MEDIA_SELECT Macro

Syntax

```
#define PRL_SCAN_CODE_MEDIA_SELECT {224,109}
```

File

PrlKeys.h

PRL_SCAN_CODE_MEDIA_STOP Macro

Syntax

```
#define PRL_SCAN_CODE_MEDIA_STOP {224,36}
```

File

PrlKeys.h

PRL_SCAN_CODE_MENU Macro

Syntax

```
#define PRL_SCAN_CODE_MENU {224,93}
```

File

PrlKeys.h

PRL_SCAN_CODE_MINUS Macro

Syntax

```
#define PRL_SCAN_CODE_MINUS {12}
```

File

PrlKeys.h

PRL_SCAN_CODE_MUHENKAN Macro

Syntax

```
#define PRL_SCAN_CODE_MUHENKAN {123}
```

File

PrlKeys.h

PRL_SCAN_CODE_MUTE Macro

Syntax

```
#define PRL_SCAN_CODE_MUTE {224,32}
```

File

PrlKeys.h

PRL_SCAN_CODE_N Macro

Syntax

```
#define PRL_SCAN_CODE_N {49}
```

File

PrlKeys.h

PRL_SCAN_CODE_NUM_LOCK Macro

Syntax

```
#define PRL_SCAN_CODE_NUM_LOCK {69}
```

File

PrlKeys.h

PRL_SCAN_CODE_O Macro

Syntax

```
#define PRL_SCAN_CODE_O {24}
```

File

PrlKeys.h

PRL_SCAN_CODE_P Macro

Syntax

```
#define PRL_SCAN_CODE_P {25}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_0 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_0 {82}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_1 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_1 {79}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_2 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_2 {80}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_3 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_3 {81}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_4 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_4 {75}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_5 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_5 {76}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_6 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_6 {77}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_7 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_7 {71}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_8 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_8 {72}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_9 Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_9 {73}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_DEL Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_DEL {83}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_ENTER Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_ENTER {224,28}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_EQUAL Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_EQUAL {89}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_MINUS Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_MINUS {74}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_PLUS Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_PLUS {78}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_SLASH Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_SLASH {224,53}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAD_STAR Macro

Syntax

```
#define PRL_SCAN_CODE_PAD_STAR {55}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAGE_DOWN Macro

Syntax

```
#define PRL_SCAN_CODE_PAGE_DOWN {224,81}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAGE_UP Macro

Syntax

```
#define PRL_SCAN_CODE_PAGE_UP {224,73}
```

File

PrlKeys.h

PRL_SCAN_CODE_PAUSE_BREAK Macro

Syntax

```
#define PRL_SCAN_CODE_PAUSE_BREAK {225,29,69,225,157,197}
```

File

PrlKeys.h

PRL_SCAN_CODE_PC9800_KEYPAD Macro

Syntax

```
#define PRL_SCAN_CODE_PC9800_KEYPAD {92}
```

File

PrlKeys.h

PRL_SCAN_CODE_PLUS Macro

Syntax

```
#define PRL_SCAN_CODE_PLUS {13}
```

File

PrlKeys.h

PRL_SCAN_CODE_PRINT_SCREEN Macro

Syntax

```
#define PRL_SCAN_CODE_PRINT_SCREEN {224,42,224,55}
```

File

PrlKeys.h

PRL_SCAN_CODE_PRINT_SCREEN2 Macro

Syntax

```
#define PRL_SCAN_CODE_PRINT_SCREEN2 {224,55}
```

File

PrlKeys.h

PRL_SCAN_CODE_Q Macro

Syntax

```
#define PRL_SCAN_CODE_Q {16}
```

File

PrlKeys.h

PRL_SCAN_CODE_QUOTE Macro

Syntax

```
#define PRL_SCAN_CODE_QUOTE {40}
```

File

PrlKeys.h

PRL_SCAN_CODE_R Macro

Syntax

```
#define PRL_SCAN_CODE_R {19}
```

File

PrlKeys.h

PRL_SCAN_CODE_RIGHT Macro

Syntax

```
#define PRL_SCAN_CODE_RIGHT {224,77}
```

File

PrlKeys.h

PRL_SCAN_CODE_RO Macro

Syntax

```
#define PRL_SCAN_CODE_RO {115}
```

File

PrlKeys.h

PRL_SCAN_CODE_S Macro

Syntax

```
#define PRL_SCAN_CODE_S {31}
```

File

PrlKeys.h

PRL_SCAN_CODE_SCROLL_LOCK Macro

Syntax

```
#define PRL_SCAN_CODE_SCROLL_LOCK {70}
```

File

PrlKeys.h

PRL_SCAN_CODE_SHIFT_LEFT Macro

Syntax

```
#define PRL_SCAN_CODE_SHIFT_LEFT {42}
```

File

PrlKeys.h

PRL_SCAN_CODE_SHIFT_RIGHT Macro

Syntax

```
#define PRL_SCAN_CODE_SHIFT_RIGHT {54}
```

File

PrlKeys.h

PRL_SCAN_CODE_SLASH Macro

Syntax

```
#define PRL_SCAN_CODE_SLASH {53}
```

File

PrlKeys.h

PRL_SCAN_CODE_SPACE Macro

Syntax

```
#define PRL_SCAN_CODE_SPACE {57}
```

File

PrlKeys.h

PRL_SCAN_CODE_SYSRQ Macro

Syntax

```
#define PRL_SCAN_CODE_SYSRQ {84}
```

File

PrlKeys.h

PRL_SCAN_CODE_SYSTEM_POWER Macro

Syntax

```
#define PRL_SCAN_CODE_SYSTEM_POWER {224,94}
```

File

PrlKeys.h

PRL_SCAN_CODE_SYSTEM_SLEEP Macro

Syntax

```
#define PRL_SCAN_CODE_SYSTEM_SLEEP {224,95}
```

File

PrlKeys.h

PRL_SCAN_CODE_SYSTEM_WAKE Macro

Syntax

```
#define PRL_SCAN_CODE_SYSTEM_WAKE {224,99}
```

File

PrlKeys.h

PRL_SCAN_CODE_T Macro

Syntax

```
#define PRL_SCAN_CODE_T {20}
```

File

PrlKeys.h

PRL_SCAN_CODE_TAB Macro

Syntax

```
#define PRL_SCAN_CODE_TAB {15}
```

File

PrlKeys.h

PRL_SCAN_CODE_TILDA Macro

Syntax

```
#define PRL_SCAN_CODE_TILDA {41}
```

File

PrlKeys.h

PRL_SCAN_CODE_U Macro

Syntax

```
#define PRL_SCAN_CODE_U {22}
```

File

PrlKeys.h

PRL_SCAN_CODE_UP Macro

Syntax

```
#define PRL_SCAN_CODE_UP {224,72}
```

File

PrlKeys.h

PRL_SCAN_CODE_V Macro

Syntax

```
#define PRL_SCAN_CODE_V {47}
```

File

PrlKeys.h

PRL_SCAN_CODE_VOLUME_DOWN Macro

Syntax

```
#define PRL_SCAN_CODE_VOLUME_DOWN {224,46}
```

File

PrlKeys.h

PRL_SCAN_CODE_VOLUME_UP Macro

Syntax

```
#define PRL_SCAN_CODE_VOLUME_UP {224,48}
```

File

PrlKeys.h

PRL_SCAN_CODE_W Macro

Syntax

```
#define PRL_SCAN_CODE_W {17}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_BACK Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_BACK {224,106}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_FAVORITES Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_FAVORITES {224,102}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_FORWARD Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_FORWARD {224,105}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_HOME Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_HOME {224,50}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_REFRESH Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_REFRESH {224,103}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_SEARCH Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_SEARCH {224,101}
```

File

PrlKeys.h

PRL_SCAN_CODE_WWW_STOP Macro

Syntax

```
#define PRL_SCAN_CODE_WWW_STOP {224,104}
```

File

PrlKeys.h

PRL_SCAN_CODE_X Macro

Syntax

```
#define PRL_SCAN_CODE_X {45}
```

File

PrlKeys.h

PRL_SCAN_CODE_Y Macro

Syntax

```
#define PRL_SCAN_CODE_Y {21}
```

File

PrlKeys.h

PRL_SCAN_CODE_YEN Macro

Syntax

```
#define PRL_SCAN_CODE_YEN {125}
```

File

PrlKeys.h

PRL_SCAN_CODE_Z Macro

Syntax

```
#define PRL_SCAN_CODE_Z {44}
```

File

PrlKeys.h

PRL_SCAN_CODE_ZENKAKU_HANKAKU Macro

Syntax

```
#define PRL_SCAN_CODE_ZENKAKU_HANKAKU {118}
```

File

PrlKeys.h

PRL_STATE Macros

Macros

Name	Description
PRL_STATE_CALLBACK_INVALID (p. 1562)	Invalid callback
PRL_STATE_CORE_FIRST_COMPLETE_CALLBACK (p. 1562)	Core callback completed
PRL_STATE_CORE_FULL_COMPLETE_CALLBACK (p. 1562)	Operation fully completed
PRL_STATE_CORE_START_DISK_PROCESS_CALLBACK (p. 1562)	Disk operation started
PRL_STATE_DISK_CALLBACK (p. 1563)	Ordinary callback that disk process going.
PRL_STATE_DISK_CALLBACK_STAGE_CPL (p. 1563)	At remove all states this callback indicates that one more stage complete in states removal
PRL_STATE_PROGRESS_COMPLETED (p. 1563)	

PRL_STATE_PROGRESS_MAX (p. 1563)	
PRL_STATE_PROGRESS_MIN (p. 1563)	States progress definition

PRL_STATE_CALLBACK_INVALID Macro

Invalid callback

Syntax

```
#define PRL_STATE_CALLBACK_INVALID 0x1005
```

File

PrlDisk.h

PRL_STATE_CORE_FIRST_COMPLETE_CALLBACK Macro

Core callback completed

Syntax

```
#define PRL_STATE_CORE_FIRST_COMPLETE_CALLBACK 0x1001
```

File

PrlDisk.h

PRL_STATE_CORE_FULL_COMPLETE_CALLBACK Macro

Operation fully completed

Syntax

```
#define PRL_STATE_CORE_FULL_COMPLETE_CALLBACK 0x1002
```

File

PrlDisk.h

PRL_STATE_CORE_START_DISK_PROCESS_CALLBACK Macro

Disk operation started

Syntax

```
#define PRL_STATE_CORE_START_DISK_PROCESS_CALLBACK 0x1003
```

File

PrlDisk.h

PRL_STATE_DISK_CALLBACK Macro

Ordinary callback that disk process going.

Syntax

```
#define PRL_STATE_DISK_CALLBACK 0x1000
```

File

PrlDisk.h

PRL_STATE_DISK_CALLBACK_STAGE_CPL Macro

At remove all states this callback indicates that one more stage complete in states removal

Syntax

```
#define PRL_STATE_DISK_CALLBACK_STAGE_CPL 0x1004
```

File

PrlDisk.h

PRL_STATE_PROGRESS_COMPLETED Macro

Syntax

```
#define PRL_STATE_PROGRESS_COMPLETED 1005
```

File

PrlDisk.h

PRL_STATE_PROGRESS_MAX Macro

Syntax

```
#define PRL_STATE_PROGRESS_MAX 1000
```

File

PrlDisk.h

PRL_STATE_PROGRESS_MIN Macro

States progress definition

Syntax

```
#define PRL_STATE_PROGRESS_MIN 0
```

File

PrlDisk.h

PRL_WARN Macros

Macros

Name	Description
PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND (p. 1565)	PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND is keep only for compatibility with previous versions of dispatcher (PSBM)
PRL_WARN_BACKUP_GUEST_SYNCHRONIZATION_FAILED (p. 1565)	
PRL_WARN_BACKUP_GUEST_UNABLE_TO_SYNCHRONIZE (p. 1566)	
PRL_WARN_BACKUP_HAS_NOT_FULL_BACKUP (p. 1566)	
PRL_WARN_BACKUP_NON_IMAGE_HDD (p. 1566)	
PRL_WARN_BACKUP_SP_GUEST_SYNCHRONIZATION_FAILED (p. 1566)	
PRL_WARN_CANNOT_MOUNT_BACK_PARTITION (p. 1567)	
PRL_WARN_CANT_CONNECT_VOLUME (p. 1567)	
PRL_WARN_COMPACT_DESTRUCTIVE (p. 1567)	
PRL_WARN_COMPACT_THROUGH_COPY (p. 1567)	
PRL_WARN_DELETE_NO_CONFIG_UNREGISTER (p. 1567)	
PRL_WARN_DEV_USB_CONNECT_FAILED (p. 1568)	
PRL_WARN_DEV_USB_OPEN_FAILED (p. 1568)	
PRL_WARN_FAILED_TO_START_VNC_SERVER (p. 1568)	
PRL_WARN_FAT32_FILE_OVERRUN (p. 1568)	
PRL_WARN_GOING_TO_TAKE_SMART_GUARD_SNAPSHOT (p. 1569)	
PRL_WARN_HDD_ERROR (p. 1569)	
PRL_WARN_IMAGE_IS_STOLEN (p. 1569)	
PRL_WARN_LAN_NOT_CONNECTED (p. 1569)	
PRL_WARN_LICENSE_RESTRICTED_RKU (p. 1570)	
PRL_WARN_LINK_TO_OUTSIDED_HDD_WILL_BE_DROPPED_AFTER_CLONE (p. 1570)	
PRL_WARN_NESTED_VIRT_NOT_ENABLED (p. 1570)	
PRL_WARN_NO_SHARED_NETWORK_FOR_OFFLINE_MANAGEMENT (p. 1570)	
PRL_WARN_NOT_SYMMETRIC_CPU (p. 1571)	
PRL_WARN_OUTSIDED_HDD_WILL_BE_ONLY_LINKED_AFTER_CLONE (p. 1571)	

PRL_WARN_PREPARE_FOR_HIBERNATE_UNABLE_SUSPEND_DO_STOP (p. 1571)	
PRL_WARN_TARGET_HOST_DISK_IS_FULL (p. 1571)	
PRL_WARN_TOO_LOW_HDD_SIZE (p. 1572)	
PRL_WARN_UNABLE_OPEN_DEVICE (p. 1572)	
PRL_WARN_UNABLE_OPEN_DEVICE_NAME (p. 1572)	
PRL_WARN_UNABLE_OPEN_DEVICE_ON_START_VM (p. 1572)	
PRL_WARN_UNKNOWN_HDD_REQUEST_TYPE (p. 1572)	
PRL_WARN_VERBOSE_LOGGING (p. 1573)	
PRL_WARN_VM_BOOTCAMP_MODE (p. 1573)	
PRL_WARN_VM_DISCONNECT_SATA_HDD (p. 1573)	
PRL_WARN_VM_DVD_DISCONNECT_LOCKED_DEV (p. 1573)	
PRL_WARN_VM_DVD_DISCONNECTED_BY_GUEST (p. 1574)	
PRL_WARN_VM_PD3_COMPAT_MODE (p. 1574)	
PRL_WARN_VM_PD3_COMPAT_NO_MOUSE (p. 1574)	
PRL_WARN_VM_SYNC_DEFAULT_PRINTER_DISABLED (p. 1574)	
PRL_WARN_WINDOWS_UPDATE_WORKING (p. 1574)	

PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND Macro

PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND is keep only for compatibility with previous versions of dispatcher (PSBM)

Syntax

```
#define PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND PRL_RESULT_DECLARE_ERROR(37025)
```

File

PrlErrorsValues.h

PRL_WARN_BACKUP_GUEST_SYNCHRONIZATION_FAILED Macro

Syntax

```
#define PRL_WARN_BACKUP_GUEST_SYNCHRONIZATION_FAILED PRL_RESULT_DECLARE_SUCCESS(37020)
```

File

PrlErrorsValues.h

PRL_WARN_BACKUP_GUEST_UNABLE_TO_SYNCHRONIZE Macro

Syntax

```
#define PRL_WARN_BACKUP_GUEST_UNABLE_TO_SYNCHRONIZE PRL_RESULT_DECLARE_SUCCESS(37029)
```

File

PrlErrorsValues.h

PRL_WARN_BACKUP_HAS_NOT_FULL_BACKUP Macro

Syntax

```
#define PRL_WARN_BACKUP_HAS_NOT_FULL_BACKUP PRL_RESULT_DECLARE_SUCCESS(37017)
```

File

PrlErrorsValues.h

PRL_WARN_BACKUP_NON_IMAGE_HDD Macro

Syntax

```
#define PRL_WARN_BACKUP_NON_IMAGE_HDD PRL_RESULT_DECLARE_SUCCESS(37037)
```

File

PrlErrorsValues.h

PRL_WARN_BACKUP_SP_GUEST_SYNCHRONIZATION_FAILED Macro

Syntax

```
#define PRL_WARN_BACKUP_SP_GUEST_SYNCHRONIZATION_FAILED PRL_RESULT_DECLARE_SUCCESS(37026)
```

File

PrlErrorsValues.h

PRL_WARN_CANNOT_MOUNT_BACK_PARTITION Macro

Syntax

```
#define PRL_WARN_CANNOT_MOUNT_BACK_PARTITION PRL_RESULT_DECLARE_ERROR(501)
```

File

PrlErrorsValues.h

PRL_WARN_CANT_CONNECT_VOLUME Macro

Syntax

```
#define PRL_WARN_CANT_CONNECT_VOLUME PRL_RESULT_DECLARE_ERROR(458)
```

File

PrlErrorsValues.h

PRL_WARN_COMPACT_DESTRUCTIVE Macro

Syntax

```
#define PRL_WARN_COMPACT_DESTRUCTIVE PRL_RESULT_DECLARE_ERROR(28002)
```

File

PrlErrorsValues.h

PRL_WARN_COMPACT_THROUGH_COPY Macro

Syntax

```
#define PRL_WARN_COMPACT_THROUGH_COPY PRL_RESULT_DECLARE_ERROR(28001)
```

File

PrlErrorsValues.h

PRL_WARN_DELETE_NO_CONFIG_UNREGISTER Macro

Syntax

```
#define PRL_WARN_DELETE_NO_CONFIG_UNREGISTER PRL_RESULT_DECLARE_SUCCESS(582)
```

File

PrlErrorsValues.h

PRL_WARN_DEV_USB_CONNECT_FAILED Macro

Syntax

```
#define PRL_WARN_DEV_USB_CONNECT_FAILED PRL_RESULT_DECLARE_ERROR(471)
```

File

PrlErrorsValues.h

PRL_WARN_DEV_USB_OPEN_FAILED Macro

Syntax

```
#define PRL_WARN_DEV_USB_OPEN_FAILED PRL_RESULT_DECLARE_ERROR(474)
```

File

PrlErrorsValues.h

PRL_WARN_FAILED_TO_START_VNC_SERVER Macro

Syntax

```
#define PRL_WARN_FAILED_TO_START_VNC_SERVER PRL_RESULT_DECLARE_SUCCESS(35006)
```

File

PrlErrorsValues.h

PRL_WARN_FAT32_FILE_OVERRUN Macro

Syntax

```
#define PRL_WARN_FAT32_FILE_OVERRUN PRL_RESULT_DECLARE_ERROR(267)
```

File

PrlErrorsValues.h

PRL_WARN_GOING_TO_TAKE_SMART_GUARD_SNAPSHOT Macro

Syntax

```
#define PRL_WARN_GOING_TO_TAKE_SMART_GUARD_SNAPSHOT PRL_RESULT_DECLARE_ERROR(446)
```

File

PrlErrorsValues.h

PRL_WARN_HDD_ERROR Macro

Syntax

```
#define PRL_WARN_HDD_ERROR PRL_RESULT_DECLARE_ERROR(268)
```

File

PrlErrorsValues.h

PRL_WARN_IMAGE_IS_STOLEN Macro

Syntax

```
#define PRL_WARN_IMAGE_IS_STOLEN PRL_RESULT_DECLARE_ERROR(264)
```

File

PrlErrorsValues.h

PRL_WARN_LAN_NOT_CONNECTED Macro

Syntax

```
#define PRL_WARN_LAN_NOT_CONNECTED PRL_RESULT_DECLARE_ERROR(261)
```

File

PrlErrorsValues.h

PRL_WARN_LICENSE_RESTRICTED_RKU Macro

Syntax

```
#define PRL_WARN_LICENSE_RESTRICTED_RKU PRL_RESULT_DECLARE_ERROR(11069)
```

File

PrlErrorsValues.h

PRL_WARN_LINK_TO_OUTSIDED_HDD_WILL_BE_DROPPED_AFTER_CLONE Macro

Syntax

```
#define PRL_WARN_LINK_TO_OUTSIDED_HDD_WILL_BE_DROPPED_AFTER_CLONE PRL_RESULT_DECLARE_ERROR(531)
```

File

PrlErrorsValues.h

PRL_WARN_NESTED_VIRT_NOT_ENABLED Macro

Syntax

```
#define PRL_WARN_NESTED_VIRT_NOT_ENABLED PRL_RESULT_DECLARE_SUCCESS(583)
```

File

PrlErrorsValues.h

PRL_WARN_NO_SHARED_NETWORK_FOR_OFFLINE_MANAGEMENT Macro

Syntax

```
#define PRL_WARN_NO_SHARED_NETWORK_FOR_OFFLINE_MANAGEMENT PRL_RESULT_DECLARE_SUCCESS(27771)
```

File

PrlErrorsValues.h

PRL_WARN_NOT_SYMMETRIC_CPU Macro

Syntax

```
#define PRL_WARN_NOT_SYMMETRIC_CPU PRL_RESULT_DECLARE_ERROR(470)
```

File

PrlErrorsValues.h

PRL_WARN_OUTSIDED_HDD_WILL_BE_ONLY_LINKED_AFTER_CLONE Macro

Syntax

```
#define PRL_WARN_OUTSIDED_HDD_WILL_BE_ONLY_LINKED_AFTER_CLONE PRL_RESULT_DECLARE_ERROR(532)
```

File

PrlErrorsValues.h

PRL_WARN_PREPARE_FOR_HIBERNATE_UNABLE_SUSPEND_DO_STOP Macro

Syntax

```
#define PRL_WARN_PREPARE_FOR_HIBERNATE_UNABLE_SUSPEND_DO_STOP PRL_RESULT_DECLARE_SUCCESS(482)
```

File

PrlErrorsValues.h

PRL_WARN_TARGET_HOST_DISK_IS_FULL Macro

Syntax

```
#define PRL_WARN_TARGET_HOST_DISK_IS_FULL PRL_RESULT_DECLARE_ERROR(265)
```

File

PrlErrorsValues.h

PRL_WARN_TOO_LOW_HDD_SIZE Macro

Syntax

```
#define PRL_WARN_TOO_LOW_HDD_SIZE PRL_RESULT_DECLARE_ERROR(289)
```

File

PrlErrorsValues.h

PRL_WARN_UNABLE_OPEN_DEVICE Macro

Syntax

```
#define PRL_WARN_UNABLE_OPEN_DEVICE PRL_RESULT_DECLARE_ERROR(263)
```

File

PrlErrorsValues.h

PRL_WARN_UNABLE_OPEN_DEVICE_NAME Macro

Syntax

```
#define PRL_WARN_UNABLE_OPEN_DEVICE_NAME PRL_RESULT_DECLARE_ERROR(262)
```

File

PrlErrorsValues.h

PRL_WARN_UNABLE_OPEN_DEVICE_ON_START_VM Macro

Syntax

```
#define PRL_WARN_UNABLE_OPEN_DEVICE_ON_START_VM PRL_RESULT_DECLARE_ERROR(9002)
```

File

PrlErrorsValues.h

PRL_WARN_UNKNOWN_HDD_REQUEST_TYPE Macro

Syntax

```
#define PRL_WARN_UNKNOWN_HDD_REQUEST_TYPE PRL_RESULT_DECLARE_ERROR(266)
```

File

PrlErrorsValues.h

PRL_WARN_VERBOSE_LOGGING Macro

Syntax

```
#define PRL_WARN_VERBOSE_LOGGING PRL_RESULT_DECLARE_SUCCESS(495)
```

File

PrlErrorsValues.h

PRL_WARN_VM_BOOTCAMP_MODE Macro

Syntax

```
#define PRL_WARN_VM_BOOTCAMP_MODE PRL_RESULT_DECLARE_ERROR(1004)
```

File

PrlErrorsValues.h

PRL_WARN_VM_DISCONNECT_SATA_HDD Macro

Syntax

```
#define PRL_WARN_VM_DISCONNECT_SATA_HDD PRL_RESULT_DECLARE_ERROR(13048)
```

File

PrlErrorsValues.h

PRL_WARN_VM_DVD_DISCONNECT_LOCKED_DEV Macro

Syntax

```
#define PRL_WARN_VM_DVD_DISCONNECT_LOCKED_DEV PRL_RESULT_DECLARE_ERROR(433)
```

File

PrlErrorsValues.h

PRL_WARN_VM_DVD_DISCONNECTED_BY_GUEST Macro

Syntax

```
#define PRL_WARN_VM_DVD_DISCONNECTED_BY_GUEST PRL_RESULT_DECLARE_ERROR(448)
```

File

PrlErrorsValues.h

PRL_WARN_VM_PD3_COMPAT_MODE Macro

Syntax

```
#define PRL_WARN_VM_PD3_COMPAT_MODE PRL_RESULT_DECLARE_ERROR(1003)
```

File

PrlErrorsValues.h

PRL_WARN_VM_PD3_COMPAT_NO_MOUSE Macro

Syntax

```
#define PRL_WARN_VM_PD3_COMPAT_NO_MOUSE PRL_RESULT_DECLARE_ERROR(1005)
```

File

PrlErrorsValues.h

PRL_WARN_VM_SYNC_DEFAULT_PRINTER_DISABLED Macro

Syntax

```
#define PRL_WARN_VM_SYNC_DEFAULT_PRINTER_DISABLED PRL_RESULT_DECLARE_ERROR(545)
```

File

PrlErrorsValues.h

PRL_WARN_WINDOWS_UPDATE_WORKING Macro

Syntax

```
#define PRL_WARN_WINDOWS_UPDATE_WORKING PRL_RESULT_DECLARE_ERROR(563)
```

File

PrlErrorsValues.h

PRL_WRN Macros

Macros

Name	Description
PRL_WRN_IT_CANT_RESIZE_LDM_DISK (p. 1575)	
PRL_WRN_IT_CANT_RESIZE_VOLUME (p. 1575)	
PRL_WRN_IT_EMPTY_MBR (p. 1575)	
PRL_WRN_IT_FIRST_WARNING (p. 1576)	vi disk tool errors +++ ALIGNED VALUE TO 46000
PRL_WRN_IT_VOLUME_RESIZING (p. 1576)	

PRL_WRN_IT_CANT_RESIZE_LDM_DISK Macro

Syntax

```
#define PRL_WRN_IT_CANT_RESIZE_LDM_DISK PRL_RESULT_DECLARE_ERROR(46004)
```

File

PrlErrorsValues.h

PRL_WRN_IT_CANT_RESIZE_VOLUME Macro

Syntax

```
#define PRL_WRN_IT_CANT_RESIZE_VOLUME PRL_RESULT_DECLARE_ERROR(46003)
```

File

PrlErrorsValues.h

PRL_WRN_IT_EMPTY_MBR Macro

Syntax

```
#define PRL_WRN_IT_EMPTY_MBR PRL_RESULT_DECLARE_ERROR(46001)
```

File

PrlErrorsValues.h

PRL_WRN_IT_FIRST_WARNING Macro

vi disk tool errors +++ ALIGNED VALUE TO 46000

Syntax

```
#define PRL_WRN_IT_FIRST_WARNING PRL_RESULT_DECLARE_ERROR(46000)
```

File

PrlErrorsValues.h

PRL_WRN_IT_VOLUME_RESIZING Macro

Syntax

```
#define PRL_WRN_IT_VOLUME_RESIZING PRL_RESULT_DECLARE_ERROR(46002)
```

File

PrlErrorsValues.h

PVS_GUEST Macros

Macros

Name	Description
PVS_GUEST_ARCH_I386 (p. 1581)	
PVS_GUEST_ARCH_OTHER (p. 1581)	Autodetected OS Architecture
PVS_GUEST_ARCH_X86_64 (p. 1582)	
PVS_GUEST_TYPE_ANDROID (p. 1582)	
PVS_GUEST_TYPE_CHROMEOS (p. 1582)	
PVS_GUEST_TYPE_FREEBSD (p. 1582)	
PVS_GUEST_TYPE_LINUX (p. 1582)	
PVS_GUEST_TYPE_MACOS (p. 1583)	Guest OS types enumeration
PVS_GUEST_TYPE_MSDOS (p. 1583)	
PVS_GUEST_TYPE_NAME_ANDROID (p. 1583)	
PVS_GUEST_TYPE_NAME_CHROMEOS (p. 1583)	
PVS_GUEST_TYPE_NAME_FREEBSD (p. 1583)	
PVS_GUEST_TYPE_NAME_LINUX (p. 1584)	
PVS_GUEST_TYPE_NAME_MACOS (p. 1584)	Guest OS type names
PVS_GUEST_TYPE_NAME_MSDOS (p. 1584)	
PVS_GUEST_TYPE_NAME_NETWARE (p. 1584)	
PVS_GUEST_TYPE_NAME_OS2 (p. 1585)	

PVS_GUEST_TYPE_NAME_OTHER (p. 1585)	
PVS_GUEST_TYPE_NAME_SOLARIS (p. 1585)	
PVS_GUEST_TYPE_NAME_WINDOWS (p. 1585)	
PVS_GUEST_TYPE_NETWARE (p. 1585)	
PVS_GUEST_TYPE_OS2 (p. 1586)	
PVS_GUEST_TYPE_OTHER (p. 1586)	
PVS_GUEST_TYPE_SOLARIS (p. 1586)	
PVS_GUEST_TYPE_WINDOWS (p. 1586)	
PVS_GUEST_VER_ANDROID_2_x (p. 1586)	
PVS_GUEST_VER_ANDROID_3_x (p. 1587)	
PVS_GUEST_VER_ANDROID_4_0 (p. 1587)	
PVS_GUEST_VER_ANDROID_LAST (p. 1587)	
PVS_GUEST_VER_ANDROID_OTHER (p. 1587)	
PVS_GUEST_VER_BSD_4X (p. 1587)	
PVS_GUEST_VER_BSD_5X (p. 1588)	
PVS_GUEST_VER_BSD_6X (p. 1588)	
PVS_GUEST_VER_BSD_7X (p. 1588)	
PVS_GUEST_VER_BSD_8X (p. 1588)	
PVS_GUEST_VER_BSD_LAST (p. 1588)	Add here a new OS version and assign it to PVS_GUEST_VER_BSD_LAST
PVS_GUEST_VER_BSD_OTHER (p. 1589)	
PVS_GUEST_VER_CHROMEOS_1x (p. 1589)	
PVS_GUEST_VER_CHROMEOS_LAST (p. 1589)	Add here a new OS version and assign it to PVS_GUEST_VER_MACOS_LAST (p. 1596)
PVS_GUEST_VER_CHROMEOS_OTHER (p. 1589)	
PVS_GUEST_VER_DOS_LAST (p. 1590)	Add here a new OS version and assign it to PVS_GUEST_VER_DOS_LAST
PVS_GUEST_VER_DOS_MS622 (p. 1590)	
PVS_GUEST_VER_DOS_OTHER (p. 1590)	
PVS_GUEST_VER_LIN_CENTOS (p. 1590)	
PVS_GUEST_VER_LIN_CENTOS_7 (p. 1590)	
PVS_GUEST_VER_LIN_CLOUDLINUX (p. 1591)	
PVS_GUEST_VER_LIN_CLOUDLINUX_7 (p. 1591)	
PVS_GUEST_VER_LIN_DEBIAN (p. 1591)	
PVS_GUEST_VER_LIN_FEDORA (p. 1591)	
PVS_GUEST_VER_LIN_FEDORA_5 (p. 1592)	
PVS_GUEST_VER_LIN_KRNL_24 (p. 1592)	
PVS_GUEST_VER_LIN_KRNL_26 (p. 1592)	
PVS_GUEST_VER_LIN_LAST (p. 1592)	Add here a new OS version and assign it to PVS_GUEST_VER_LIN_LAST
PVS_GUEST_VER_LIN_MAGEIA (p. 1592)	
PVS_GUEST_VER_LIN_MANDRAKE (p. 1593)	
PVS_GUEST_VER_LIN_MINT (p. 1593)	
PVS_GUEST_VER_LIN_OPENSUSE (p. 1593)	

PVS_GUEST_VER_LIN_OTHER (p. 1593)	
PVS_GUEST_VER_LIN_PSBM (p. 1594)	
PVS_GUEST_VER_LIN_REDHAT (p. 1594)	
PVS_GUEST_VER_LIN_REDHAT_7 (p. 1594)	
PVS_GUEST_VER_LIN_RH_LEGACY (p. 1594)	
PVS_GUEST_VER_LIN_RHLES3 (p. 1594)	
PVS_GUEST_VER_LIN_SLES11 (p. 1595)	
PVS_GUEST_VER_LIN_SLES12 (p. 1595)	
PVS_GUEST_VER_LIN_SLES9 (p. 1595)	
PVS_GUEST_VER_LIN_SUSE (p. 1595)	
PVS_GUEST_VER_LIN_UBUNTU (p. 1595)	
PVS_GUEST_VER_LIN_VZLINUX (p. 1596)	
PVS_GUEST_VER_LIN_VZLINUX_7 (p. 1596)	
PVS_GUEST_VER_LIN_XANDROS (p. 1596)	
PVS_GUEST_VER_MACOS_LAST (p. 1596)	Add here a new OS version and assign it to PVS_GUEST_VER_MACOS_LAST
PVS_GUEST_VER_MACOS_LEOPARD (p. 1596)	
PVS_GUEST_VER_MACOS_SNOW_LEOPARD (p. 1597)	
PVS_GUEST_VER_MACOS_TIGER (p. 1597)	
PVS_GUEST_VER_MACOS_UNIVERSAL (p. 1597)	
PVS_GUEST_VER_NAME_ANDROID_4_0 (p. 1597)	
PVS_GUEST_VER_NAME_ANDROID_OTHER (p. 1598)	
PVS_GUEST_VER_NAME_BSD_4X (p. 1598)	
PVS_GUEST_VER_NAME_BSD_5X (p. 1598)	
PVS_GUEST_VER_NAME_BSD_6X (p. 1598)	
PVS_GUEST_VER_NAME_BSD_7X (p. 1598)	
PVS_GUEST_VER_NAME_BSD_8X (p. 1599)	
PVS_GUEST_VER_NAME_BSD_OTHER (p. 1599)	
PVS_GUEST_VER_NAME_CHROMEOS_1x (p. 1599)	
PVS_GUEST_VER_NAME_CHROMEOS_OTHER (p. 1599)	
PVS_GUEST_VER_NAME_DOS_MS622 (p. 1599)	
PVS_GUEST_VER_NAME_DOS_OTHER (p. 1600)	
PVS_GUEST_VER_NAME_LIN_CENTOS (p. 1600)	
PVS_GUEST_VER_NAME_LIN_CLOUDLINUX (p. 1600)	
PVS_GUEST_VER_NAME_LIN_DEBIAN (p. 1600)	

PVS_GUEST_VER_NAME_LIN_FEDORA (p. 1600)	
PVS_GUEST_VER_NAME_LIN_FEDORA_5 (p. 1601)	
PVS_GUEST_VER_NAME_LIN_KRNL_24 (p. 1601)	
PVS_GUEST_VER_NAME_LIN_KRNL_26 (p. 1601)	
PVS_GUEST_VER_NAME_LIN_MAGEIA (p. 1601)	
PVS_GUEST_VER_NAME_LIN_MANDRAKE (p. 1601)	
PVS_GUEST_VER_NAME_LIN_MINT (p. 1602)	
PVS_GUEST_VER_NAME_LIN_OPENSUSE (p. 1602)	
PVS_GUEST_VER_NAME_LIN_OTHER (p. 1602)	
PVS_GUEST_VER_NAME_LIN_PSBM (p. 1602)	
PVS_GUEST_VER_NAME_LIN_REDHAT (p. 1602)	
PVS_GUEST_VER_NAME_LIN_RH_LEGACY (p. 1603)	
PVS_GUEST_VER_NAME_LIN_RHLES3 (p. 1603)	
PVS_GUEST_VER_NAME_LIN_SLES9 (p. 1603)	
PVS_GUEST_VER_NAME_LIN_SUSE (p. 1603)	
PVS_GUEST_VER_NAME_LIN_UBUNTU (p. 1603)	
PVS_GUEST_VER_NAME_LIN_VZLINUX (p. 1604)	
PVS_GUEST_VER_NAME_LIN_XANDROS (p. 1604)	
PVS_GUEST_VER_NAME_MACOS_LEOPARD (p. 1604)	
PVS_GUEST_VER_NAME_MACOS_SNOW_LEOPARD (p. 1604)	
PVS_GUEST_VER_NAME_MACOS_TIGER (p. 1604)	Guest OS version names
PVS_GUEST_VER_NAME_NET_4X (p. 1605)	
PVS_GUEST_VER_NAME_NET_5X (p. 1605)	
PVS_GUEST_VER_NAME_NET_6X (p. 1605)	
PVS_GUEST_VER_NAME_NET_OTHER (p. 1605)	
PVS_GUEST_VER_NAME_OS2_ECS11 (p. 1606)	
PVS_GUEST_VER_NAME_OS2_ECS12 (p. 1606)	
PVS_GUEST_VER_NAME_OS2_OTHER (p. 1606)	
PVS_GUEST_VER_NAME_OS2_WARP3 (p. 1606)	
PVS_GUEST_VER_NAME_OS2_WARP4 (p. 1606)	
PVS_GUEST_VER_NAME_OS2_WARP45 (p. 1607)	
PVS_GUEST_VER_NAME_OTH_OPENSTEP (p. 1607)	
PVS_GUEST_VER_NAME_OTH_OTHER (p. 1607)	
PVS_GUEST_VER_NAME_OTH_QNX (p. 1607)	

PVS_GUEST_VER_NAME_SOL_10 (p. 1607)	
PVS_GUEST_VER_NAME_SOL_11 (p. 1608)	
PVS_GUEST_VER_NAME_SOL_9 (p. 1608)	
PVS_GUEST_VER_NAME_SOL_OPEN (p. 1608)	
PVS_GUEST_VER_NAME_SOL_OTHER (p. 1608)	
PVS_GUEST_VER_NAME_WIN_2003 (p. 1608)	
PVS_GUEST_VER_NAME_WIN_2008 (p. 1609)	
PVS_GUEST_VER_NAME_WIN_2012 (p. 1609)	
PVS_GUEST_VER_NAME_WIN_2K (p. 1609)	
PVS_GUEST_VER_NAME_WIN_311 (p. 1609)	
PVS_GUEST_VER_NAME_WIN_95 (p. 1609)	
PVS_GUEST_VER_NAME_WIN_98 (p. 1610)	
PVS_GUEST_VER_NAME_WIN_ME (p. 1610)	
PVS_GUEST_VER_NAME_WIN_NT (p. 1610)	
PVS_GUEST_VER_NAME_WIN_OTHER (p. 1610)	
PVS_GUEST_VER_NAME_WIN_VISTA (p. 1610)	
PVS_GUEST_VER_NAME_WIN_WINDOWS7 (p. 1611)	
PVS_GUEST_VER_NAME_WIN_WINDOWS8 (p. 1611)	
PVS_GUEST_VER_NAME_WIN_WINDOWS8_1 (p. 1611)	
PVS_GUEST_VER_NAME_WIN_XP (p. 1611)	
PVS_GUEST_VER_NET_4X (p. 1611)	
PVS_GUEST_VER_NET_5X (p. 1612)	
PVS_GUEST_VER_NET_6X (p. 1612)	
PVS_GUEST_VER_NET_LAST (p. 1612)	Add here a new OS version and assign it to PVS_GUEST_VER_NET_LAST
PVS_GUEST_VER_NET_OTHER (p. 1612)	
PVS_GUEST_VER_OS2_ECS11 (p. 1612)	
PVS_GUEST_VER_OS2_ECS12 (p. 1613)	
PVS_GUEST_VER_OS2_LAST (p. 1613)	Add here a new OS version and assign it to PVS_GUEST_VER_OS2_LAST
PVS_GUEST_VER_OS2_OTHER (p. 1613)	
PVS_GUEST_VER_OS2_WARP3 (p. 1613)	
PVS_GUEST_VER_OS2_WARP4 (p. 1614)	
PVS_GUEST_VER_OS2_WARP45 (p. 1614)	
PVS_GUEST_VER_OTH_LAST (p. 1614)	Add here a new OS version and assign it to PVS_GUEST_VER_OTH_LAST
PVS_GUEST_VER_OTH_OPENSTEP (p. 1614)	
PVS_GUEST_VER_OTH_OTHER (p. 1614)	
PVS_GUEST_VER_OTH_QNX (p. 1615)	
PVS_GUEST_VER_SOL_10 (p. 1615)	
PVS_GUEST_VER_SOL_11 (p. 1615)	
PVS_GUEST_VER_SOL_9 (p. 1615)	
PVS_GUEST_VER_SOL_LAST (p. 1615)	Add here a new OS version and assign it to PVS_GUEST_VER_SOL_LAST
PVS_GUEST_VER_SOL_OPEN (p. 1616)	OpenSolaris

PVS_GUEST_VER_SOL_OTHER (p. 1616)	
PVS_GUEST_VER_WIN_2003 (p. 1616)	
PVS_GUEST_VER_WIN_2008 (p. 1616)	
PVS_GUEST_VER_WIN_2012 (p. 1617)	
PVS_GUEST_VER_WIN_2016 (p. 1617)	
PVS_GUEST_VER_WIN_2K (p. 1617)	
PVS_GUEST_VER_WIN_311 (p. 1617)	
PVS_GUEST_VER_WIN_95 (p. 1617)	
PVS_GUEST_VER_WIN_98 (p. 1618)	
PVS_GUEST_VER_WIN_LAST (p. 1618)	Add here a new OS version and assign it to PVS_GUEST_VER_WIN_LAST
PVS_GUEST_VER_WIN_ME (p. 1618)	
PVS_GUEST_VER_WIN_NT (p. 1618)	
PVS_GUEST_VER_WIN_OTHER (p. 1618)	
PVS_GUEST_VER_WIN_VISTA (p. 1619)	
PVS_GUEST_VER_WIN_WINDOWS10 (p. 1619)	
PVS_GUEST_VER_WIN_WINDOWS7 (p. 1619)	
PVS_GUEST_VER_WIN_WINDOWS8 (p. 1619)	
PVS_GUEST_VER_WIN_WINDOWS8_1 (p. 1620)	
PVS_GUEST_VER_WIN_XP (p. 1620)	

PVS_GUEST_ARCH_I386 Macro

Syntax

```
#define PVS_GUEST_ARCH_I386 (1)
```

File

PrlOses.h

PVS_GUEST_ARCH_OTHER Macro

Autodetected OS Architecture

Syntax

```
#define PVS_GUEST_ARCH_OTHER (0)
```

File

PrlOses.h

PVS_GUEST_ARCH_X86_64 Macro

Syntax

```
#define PVS_GUEST_ARCH_X86_64 (2)
```

File

PrlOses.h

PVS_GUEST_TYPE_ANDROID Macro

Syntax

```
#define PVS_GUEST_TYPE_ANDROID 0x10
```

File

PrlOses.h

PVS_GUEST_TYPE_CHROMEOS Macro

Syntax

```
#define PVS_GUEST_TYPE_CHROMEOS 0x0F
```

File

PrlOses.h

PVS_GUEST_TYPE_FREEBSD Macro

Syntax

```
#define PVS_GUEST_TYPE_FREEBSD 0x0A
```

File

PrlOses.h

PVS_GUEST_TYPE_LINUX Macro

Syntax

```
#define PVS_GUEST_TYPE_LINUX 0x09
```

File

PrlOses.h

PVS_GUEST_TYPE_MACOS Macro

Guest OS types enumeration

Syntax

```
#define PVS_GUEST_TYPE_MACOS 0x07
```

File

PrlOses.h

PVS_GUEST_TYPE_MSDOS Macro

Syntax

```
#define PVS_GUEST_TYPE_MSDOS 0x0C
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_ANDROID Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_ANDROID "Android"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_CHROMEOS Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_CHROMEOS "Chrome OS"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_FREEBSD Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_FREEBSD "FreeBSD"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_LINUX Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_LINUX "Linux"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_MACOS Macro

Guest OS type names

Syntax

```
#define PVS_GUEST_TYPE_NAME_MACOS "Mac OS X"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_MSDOS Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_MSDOS "MS-DOS"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_NETWARE Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_NETWARE "NetWare"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_OS2 Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_OS2 "OS/2"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_OTHER Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_OTHER "Other"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_SOLARIS Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_SOLARIS "Solaris"
```

File

PrlOses.h

PVS_GUEST_TYPE_NAME_WINDOWS Macro

Syntax

```
#define PVS_GUEST_TYPE_NAME_WINDOWS "Windows"
```

File

PrlOses.h

PVS_GUEST_TYPE_NETWARE Macro

Syntax

```
#define PVS_GUEST_TYPE_NETWARE 0x0D
```

File

PrlOses.h

PVS_GUEST_TYPE_OS2 Macro

Syntax

```
#define PVS_GUEST_TYPE_OS2 0x0B
```

File

PrlOses.h

PVS_GUEST_TYPE_OTHER Macro

Syntax

```
#define PVS_GUEST_TYPE_OTHER 0xFF
```

File

PrlOses.h

PVS_GUEST_TYPE_SOLARIS Macro

Syntax

```
#define PVS_GUEST_TYPE_SOLARIS 0x0E
```

File

PrlOses.h

PVS_GUEST_TYPE_WINDOWS Macro

Syntax

```
#define PVS_GUEST_TYPE_WINDOWS 0x08
```

File

PrlOses.h

PVS_GUEST_VER_ANDROID_2_x Macro

Syntax

```
#define PVS_GUEST_VER_ANDROID_2_x 0x1000
```

File

PrlOses.h

PVS_GUEST_VER_ANDROID_3_x Macro

Syntax

```
#define PVS_GUEST_VER_ANDROID_3_x 0x1001
```

File

PrlOses.h

PVS_GUEST_VER_ANDROID_4_0 Macro

Syntax

```
#define PVS_GUEST_VER_ANDROID_4_0 0x1002
```

File

PrlOses.h

PVS_GUEST_VER_ANDROID_LAST Macro

Syntax

```
#define PVS_GUEST_VER_ANDROID_LAST PVS_GUEST_VER_ANDROID_4_0
```

File

PrlOses.h

PVS_GUEST_VER_ANDROID_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_ANDROID_OTHER 0x10FF
```

File

PrlOses.h

PVS_GUEST_VER_BSD_4X Macro

Syntax

```
#define PVS_GUEST_VER_BSD_4X 0x0A01
```

File

PrlOses.h

PVS_GUEST_VER_BSD_5X Macro

Syntax

```
#define PVS_GUEST_VER_BSD_5X 0x0A02
```

File

PrlOses.h

PVS_GUEST_VER_BSD_6X Macro

Syntax

```
#define PVS_GUEST_VER_BSD_6X 0x0A03
```

File

PrlOses.h

PVS_GUEST_VER_BSD_7X Macro

Syntax

```
#define PVS_GUEST_VER_BSD_7X 0x0A04
```

File

PrlOses.h

PVS_GUEST_VER_BSD_8X Macro

Syntax

```
#define PVS_GUEST_VER_BSD_8X 0x0A05
```

File

PrlOses.h

PVS_GUEST_VER_BSD_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_BSD_LAST

Syntax

```
#define PVS_GUEST_VER_BSD_LAST PVS_GUEST_VER_BSD_8X
```

File

PrlOses.h

PVS_GUEST_VER_BSD_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_BSD_OTHER 0x0AFF
```

File

PrlOses.h

PVS_GUEST_VER_CHROMEOS_1x Macro

Syntax

```
#define PVS_GUEST_VER_CHROMEOS_1x 0x0F01
```

File

PrlOses.h

PVS_GUEST_VER_CHROMEOS_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_MACOS_LAST (p. 1596)

Syntax

```
#define PVS_GUEST_VER_CHROMEOS_LAST PVS_GUEST_VER_CHROMEOS_1x
```

File

PrlOses.h

PVS_GUEST_VER_CHROMEOS_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_CHROMEOS_OTHER 0x0FFF
```

File

PrlOses.h

PVS_GUEST_VER_DOS_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_DOS_LAST

Syntax

```
#define PVS_GUEST_VER_DOS_LAST PVS_GUEST_VER_DOS_MS622
```

File

PrlOses.h

PVS_GUEST_VER_DOS_MS622 Macro

Syntax

```
#define PVS_GUEST_VER_DOS_MS622 0x0C01
```

File

PrlOses.h

PVS_GUEST_VER_DOS_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_DOS_OTHER 0x0CFF
```

File

PrlOses.h

PVS_GUEST_VER_LIN_CENTOS Macro

Syntax

```
#define PVS_GUEST_VER_LIN_CENTOS 0x090D
```

File

PrlOses.h

PVS_GUEST_VER_LIN_CENTOS_7 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_CENTOS_7 0x0914
```

File

PrlOses.h

PVS_GUEST_VER_LIN_CLOUDLINUX Macro

Syntax

```
#define PVS_GUEST_VER_LIN_CLOUDLINUX 0x0919
```

File

PrlOses.h

PVS_GUEST_VER_LIN_CLOUDLINUX_7 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_CLOUDLINUX_7 0x0920
```

File

PrlOses.h

PVS_GUEST_VER_LIN_DEBIAN Macro

Syntax

```
#define PVS_GUEST_VER_LIN_DEBIAN 0x0906
```

File

PrlOses.h

PVS_GUEST_VER_LIN_FEDORA Macro

Syntax

```
#define PVS_GUEST_VER_LIN_FEDORA 0x0907
```

File

PrlOses.h

PVS_GUEST_VER_LIN_FEDORA_5 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_FEDORA_5 0x0908
```

File

PrlOses.h

PVS_GUEST_VER_LIN_KRNL_24 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_KRNL_24 0x0904
```

File

PrlOses.h

PVS_GUEST_VER_LIN_KRNL_26 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_KRNL_26 0x0905
```

File

PrlOses.h

PVS_GUEST_VER_LIN_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_LIN_LAST

Syntax

```
#define PVS_GUEST_VER_LIN_LAST PVS_GUEST_VER_LIN_CLOUDLINUX_7
```

File

PrlOses.h

PVS_GUEST_VER_LIN_MAGEIA Macro

Syntax

```
#define PVS_GUEST_VER_LIN_MAGEIA 0x0911
```

File

PrlOses.h

PVS_GUEST_VER_LIN_MANDRAKE Macro

Syntax

```
#define PVS_GUEST_VER_LIN_MANDRAKE 0x0903
```

File

PrlOses.h

PVS_GUEST_VER_LIN_MINT Macro

Syntax

```
#define PVS_GUEST_VER_LIN_MINT 0x0912
```

File

PrlOses.h

PVS_GUEST_VER_LIN_OPENSUSE Macro

Syntax

```
#define PVS_GUEST_VER_LIN_OPENSUSE 0x090F
```

File

PrlOses.h

PVS_GUEST_VER_LIN_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_LIN_OTHER 0x09FF
```

File

PrlOses.h

PVS_GUEST_VER_LIN_PSBM Macro

Syntax

```
#define PVS_GUEST_VER_LIN_PSBM 0x0910
```

File

PrlOses.h

PVS_GUEST_VER_LIN_REDHAT Macro

Syntax

```
#define PVS_GUEST_VER_LIN_REDHAT 0x0901
```

File

PrlOses.h

PVS_GUEST_VER_LIN_REDHAT_7 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_REDHAT_7 0x0913
```

File

PrlOses.h

PVS_GUEST_VER_LIN_RH_LEGACY Macro

Syntax

```
#define PVS_GUEST_VER_LIN_RH_LEGACY 0x090E
```

File

PrlOses.h

PVS_GUEST_VER_LIN_RHLES3 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_RHLES3 0x090C
```

File

PrlOses.h

PVS_GUEST_VER_LIN_SLES11 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_SLES11 0x0917
```

File

PrlOses.h

PVS_GUEST_VER_LIN_SLES12 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_SLES12 0x0918
```

File

PrlOses.h

PVS_GUEST_VER_LIN_SLES9 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_SLES9 0x090B
```

File

PrlOses.h

PVS_GUEST_VER_LIN_SUSE Macro

Syntax

```
#define PVS_GUEST_VER_LIN_SUSE 0x0902
```

File

PrlOses.h

PVS_GUEST_VER_LIN_UBUNTU Macro

Syntax

```
#define PVS_GUEST_VER_LIN_UBUNTU 0x090A
```

File

PrlOses.h

PVS_GUEST_VER_LIN_VZLINUX Macro

Syntax

```
#define PVS_GUEST_VER_LIN_VZLINUX 0x0915
```

File

PrlOses.h

PVS_GUEST_VER_LIN_VZLINUX_7 Macro

Syntax

```
#define PVS_GUEST_VER_LIN_VZLINUX_7 0x0916
```

File

PrlOses.h

PVS_GUEST_VER_LIN_XANDROS Macro

Syntax

```
#define PVS_GUEST_VER_LIN_XANDROS 0x0909
```

File

PrlOses.h

PVS_GUEST_VER_MACOS_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_MACOS_LAST

Syntax

```
#define PVS_GUEST_VER_MACOS_LAST PVS_GUEST_VER_MACOS_UNIVERSAL
```

File

PrlOses.h

PVS_GUEST_VER_MACOS_LEOPARD Macro

Syntax

```
#define PVS_GUEST_VER_MACOS_LEOPARD 0x0702
```

File

PrlOses.h

PVS_GUEST_VER_MACOS_SNOW_LEOPARD Macro

Syntax

```
#define PVS_GUEST_VER_MACOS_SNOW_LEOPARD 0x0703
```

File

PrlOses.h

PVS_GUEST_VER_MACOS_TIGER Macro

Syntax

```
#define PVS_GUEST_VER_MACOS_TIGER 0x0701
```

File

PrlOses.h

PVS_GUEST_VER_MACOS_UNIVERSAL Macro

Syntax

```
#define PVS_GUEST_VER_MACOS_UNIVERSAL PVS_GUEST_VER_MACOS_SNOW_LEOPARD
```

File

PrlOses.h

PVS_GUEST_VER_NAME_ANDROID_4_0 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_ANDROID_4_0 "Android"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_ANDROID_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_ANDROID_OTHER "Other Android"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_BSD_4X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_BSD_4X "FreeBSD 4.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_BSD_5X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_BSD_5X "FreeBSD 5.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_BSD_6X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_BSD_6X "FreeBSD 6.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_BSD_7X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_BSD_7X "FreeBSD 7.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_BSD_8X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_BSD_8X "FreeBSD 8.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_BSD_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_BSD_OTHER "Other FreeBSD"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_CHROMEOS_1x Macro

Syntax

```
#define PVS_GUEST_VER_NAME_CHROMEOS_1x "Chrome OS"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_CHROMEOS_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_CHROMEOS_OTHER "Other Chrome OS"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_DOS_MS622 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_DOS_MS622 "MS-DOS 6.22"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_DOS_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_DOS_OTHER "Other DOS"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_CENTOS Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_CENTOS "CentOS Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_CLOUDLINUX Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_CLOUDLINUX "CloudLinux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_DEBIAN Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_DEBIAN "Debian GNU/Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_FEDORA Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_FEDORA "Fedora Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_FEDORA_5 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_FEDORA_5 "Fedora Core 5 Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_KRNL_24 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_KRNL_24 "Other Linux kernel 2.4"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_KRNL_26 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_KRNL_26 "Other Linux kernel 2.6"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_MAGEIA Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_MAGEIA "Mageia Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_MANDRAKE Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_MANDRAKE "Mandriva Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_MINT Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_MINT "Mint Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_OPENSUSE Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_OPENSUSE "OpenSUSE Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_OTHER "Other Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_PSBM Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_PSBM "Virtuozzo Cloud Server"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_REDHAT Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_REDHAT "Red Hat Enterprise Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_RH_LEGACY Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_RH_LEGACY "Red Hat Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_RHLES3 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_RHLES3 "Red Hat Enterprise Server 3"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_SLES9 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_SLES9 "SUSE Linux Enterprise Server 9"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_SUSE Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_SUSE "SUSE Linux Enterprise"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_UBUNTU Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_UBUNTU "Ubuntu Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_VZLINUX Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_VZLINUX "Virtuozzo Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_LIN_XANDROS Macro

Syntax

```
#define PVS_GUEST_VER_NAME_LIN_XANDROS "Xandros Linux"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_MACOS_LEOPARD Macro

Syntax

```
#define PVS_GUEST_VER_NAME_MACOS_LEOPARD "Mac OS X"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_MACOS_SNOW_LEOPARD Macro

Syntax

```
#define PVS_GUEST_VER_NAME_MACOS_SNOW_LEOPARD "Mac OS X"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_MACOS_TIGER Macro

Guest OS version names

Syntax

```
#define PVS_GUEST_VER_NAME_MACOS_TIGER "Mac OS X"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_NET_4X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_NET_4X "NetWare 4.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_NET_5X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_NET_5X "NetWare 5.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_NET_6X Macro

Syntax

```
#define PVS_GUEST_VER_NAME_NET_6X "NetWare 6.x"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_NET_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_NET_OTHER "Other NetWare"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OS2_ECS11 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OS2_ECS11 "eComStation 1.1"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OS2_ECS12 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OS2_ECS12 "eComStation 1.2"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OS2_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OS2_OTHER "Other OS/2"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OS2_WARP3 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OS2_WARP3 "OS/2 Warp 3"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OS2_WARP4 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OS2_WARP4 "OS/2 Warp 4"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OS2_WARP45 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OS2_WARP45 "OS/2 Warp 4.5"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OTH_OPENSTEP Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OTH_OPENSTEP "OpenStep"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OTH_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OTH_OTHER "Other"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_OTH_QNX Macro

Syntax

```
#define PVS_GUEST_VER_NAME_OTH_QNX "QNX"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_SOL_10 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_SOL_10 "Solaris 10"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_SOL_11 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_SOL_11 "Solaris 11"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_SOL_9 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_SOL_9 "Solaris 9"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_SOL_OPEN Macro

Syntax

```
#define PVS_GUEST_VER_NAME_SOL_OPEN "OpenSolaris"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_SOL_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_SOL_OTHER "Other Solaris"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_2003 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_2003 "Windows Server 2003"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_2008 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_2008 "Windows Server 2008"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_2012 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_2012 "Windows Server 2012"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_2K Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_2K "Windows 2000"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_311 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_311 "Windows 3.11"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_95 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_95 "Windows 95"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_98 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_98 "Windows 98"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_ME Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_ME "Windows ME"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_NT Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_NT "Windows NT"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_OTHER "Other Windows"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_VISTA Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_VISTA "Windows Vista"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_WINDOWS7 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_WINDOWS7 "Windows 7"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_WINDOWS8 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_WINDOWS8 "Windows 8"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_WINDOWS8_1 Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_WINDOWS8_1 "Windows 8.1"
```

File

PrlOses.h

PVS_GUEST_VER_NAME_WIN_XP Macro

Syntax

```
#define PVS_GUEST_VER_NAME_WIN_XP "Windows XP"
```

File

PrlOses.h

PVS_GUEST_VER_NET_4X Macro

Syntax

```
#define PVS_GUEST_VER_NET_4X 0x0D01
```

File

PrlOses.h

PVS_GUEST_VER_NET_5X Macro

Syntax

```
#define PVS_GUEST_VER_NET_5X 0x0D02
```

File

PrlOses.h

PVS_GUEST_VER_NET_6X Macro

Syntax

```
#define PVS_GUEST_VER_NET_6X 0x0D03
```

File

PrlOses.h

PVS_GUEST_VER_NET_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_NET_LAST

Syntax

```
#define PVS_GUEST_VER_NET_LAST PVS_GUEST_VER_NET_6X
```

File

PrlOses.h

PVS_GUEST_VER_NET_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_NET_OTHER 0x0DFF
```

File

PrlOses.h

PVS_GUEST_VER_OS2_ECS11 Macro

Syntax

```
#define PVS_GUEST_VER_OS2_ECS11 0x0B04
```

File

PrlOses.h

PVS_GUEST_VER_OS2_ECS12 Macro

Syntax

```
#define PVS_GUEST_VER_OS2_ECS12 0x0B05
```

File

PrlOses.h

PVS_GUEST_VER_OS2_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_OS2_LAST

Syntax

```
#define PVS_GUEST_VER_OS2_LAST PVS_GUEST_VER_OS2_ECS12
```

File

PrlOses.h

PVS_GUEST_VER_OS2_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_OS2_OTHER 0x0BFF
```

File

PrlOses.h

PVS_GUEST_VER_OS2_WARP3 Macro

Syntax

```
#define PVS_GUEST_VER_OS2_WARP3 0x0B01
```

File

PrlOses.h

PVS_GUEST_VER_OS2_WARP4 Macro

Syntax

```
#define PVS_GUEST_VER_OS2_WARP4 0x0B02
```

File

PrlOses.h

PVS_GUEST_VER_OS2_WARP45 Macro

Syntax

```
#define PVS_GUEST_VER_OS2_WARP45 0x0B03
```

File

PrlOses.h

PVS_GUEST_VER_OTH_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_OTH_LAST

Syntax

```
#define PVS_GUEST_VER_OTH_LAST PVS_GUEST_VER_OTH_OPENSTEP
```

File

PrlOses.h

PVS_GUEST_VER_OTH_OPENSTEP Macro

Syntax

```
#define PVS_GUEST_VER_OTH_OPENSTEP 0xFF02
```

File

PrlOses.h

PVS_GUEST_VER_OTH_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_OTH_OTHER 0xFFFF
```

File

PrlOses.h

PVS_GUEST_VER_OTH_QNX Macro

Syntax

```
#define PVS_GUEST_VER_OTH_QNX 0xFF01
```

File

PrlOses.h

PVS_GUEST_VER_SOL_10 Macro

Syntax

```
#define PVS_GUEST_VER_SOL_10 0x0E02
```

File

PrlOses.h

PVS_GUEST_VER_SOL_11 Macro

Syntax

```
#define PVS_GUEST_VER_SOL_11 0x0E03
```

File

PrlOses.h

PVS_GUEST_VER_SOL_9 Macro

Syntax

```
#define PVS_GUEST_VER_SOL_9 0x0E01
```

File

PrlOses.h

PVS_GUEST_VER_SOL_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_SOL_LAST

Syntax

```
#define PVS_GUEST_VER_SOL_LAST PVS_GUEST_VER_SOL_OPEN
```

File

PrlOses.h

PVS_GUEST_VER_SOL_OPEN Macro

OpenSolaris

Syntax

```
#define PVS_GUEST_VER_SOL_OPEN 0x0E04 // OpenSolaris
```

File

PrlOses.h

PVS_GUEST_VER_SOL_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_SOL_OTHER 0x0EFF
```

File

PrlOses.h

PVS_GUEST_VER_WIN_2003 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_2003 0x0808
```

File

PrlOses.h

PVS_GUEST_VER_WIN_2008 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_2008 0x080A
```

File

PrlOses.h

PVS_GUEST_VER_WIN_2012 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_2012 0x080D
```

File

PrlOses.h

PVS_GUEST_VER_WIN_2016 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_2016 0x0810
```

File

PrlOses.h

PVS_GUEST_VER_WIN_2K Macro

Syntax

```
#define PVS_GUEST_VER_WIN_2K 0x0806
```

File

PrlOses.h

PVS_GUEST_VER_WIN_311 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_311 0x0801
```

File

PrlOses.h

PVS_GUEST_VER_WIN_95 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_95 0x0802
```

File

PrlOses.h

PVS_GUEST_VER_WIN_98 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_98 0x0803
```

File

PrlOses.h

PVS_GUEST_VER_WIN_LAST Macro

Add here a new OS version and assign it to PVS_GUEST_VER_WIN_LAST

Syntax

```
#define PVS_GUEST_VER_WIN_LAST PVS_GUEST_VER_WIN_2016
```

File

PrlOses.h

PVS_GUEST_VER_WIN_ME Macro

Syntax

```
#define PVS_GUEST_VER_WIN_ME 0x0804
```

File

PrlOses.h

PVS_GUEST_VER_WIN_NT Macro

Syntax

```
#define PVS_GUEST_VER_WIN_NT 0x0805
```

File

PrlOses.h

PVS_GUEST_VER_WIN_OTHER Macro

Syntax

```
#define PVS_GUEST_VER_WIN_OTHER 0x08FF
```

File

PrlOses.h

PVS_GUEST_VER_WIN_VISTA Macro

Syntax

```
#define PVS_GUEST_VER_WIN_VISTA 0x0809
```

File

PrlOses.h

PVS_GUEST_VER_WIN_WINDOWS10 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_WINDOWS10 0x080F
```

File

PrlOses.h

PVS_GUEST_VER_WIN_WINDOWS7 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_WINDOWS7 0x080B
```

File

PrlOses.h

PVS_GUEST_VER_WIN_WINDOWS8 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_WINDOWS8 0x080C
```

File

PrlOses.h

PVS_GUEST_VER_WIN_WINDOWS8_1 Macro

Syntax

```
#define PVS_GUEST_VER_WIN_WINDOWS8_1 0x080E
```

File

PrlOses.h

PVS_GUEST_VER_WIN_XP Macro

Syntax

```
#define PVS_GUEST_VER_WIN_XP 0x0807
```

File

PrlOses.h

Other PRL Macros

Macros

Name	Description
PRL_ASYNC_METHOD (p. 1622)	
PRL_ASYNC_METHOD_PTR (p. 1622)	
PRL_BSD_PARTITION (p. 1622)	Free BSD partition.
PRL_CALL (p. 1623)	Default calling conversion for every supported platform.
PRL_CHECKED_DISK_INVALID (p. 1623)	
PRL_CHECKED_DISK_OLD_VERSION (p. 1623)	
PRL_CHECKED_DISK_VALID (p. 1623)	• Disk validation errors +++ ALIGNED VALUE TO 25000
PRL_CURRENT_GUEST_OS_SESSION (p. 1624)	
PRL_DATA_PARTITION (p. 1624)	Special handle for data partition from GPT (can be linux, windows, any type).
PRL_DISK_NO_IMAGE_CREATION (p. 1624)	So not create image when creating disk, just put reference to file in XML
PRL_DISPATCHER_LISTEN_PORT (p. 1624)	
PRL_DISPATCHER_LISTEN_PORT_PMOBILE (p. 1624)	
PRL_EFI_PARTITION (p. 1625)	
PRL_EFI_PARTITION_GPT (p. 1625)	EFI system partition
PRL_EMPTY_PARTITION (p. 1625)	

PRL_EXTERN (p. 1625)	Default extern "C" if cplusplus is used
PRL_FAILED (p. 1626)	
PRL_FALSE (p. 1626)	
PRL_HOST_CPU_TIME_PTRN (p. 1626)	Host time change since the last query, in microseconds
PRL_IDE_READ_REQUESTS_PTRN (p. 1626)	Total count of read requests to IDE controller
PRL_IDE_READ_TOTAL_PTRN (p. 1626)	Total count of read bytes through IDE controller
PRL_IDE_WRITE_REQUESTS_PTRN (p. 1627)	Total count of write requests to IDE controller
PRL_IDE_WRITE_TOTAL_PTRN (p. 1627)	Total count of write bytes through IDE controller
PRL_INFO_VM_MIGRATE_STORAGE_IS_SHARED (p. 1627)	
PRL_INVALID_HANDLE (p. 1627)	
PRL_INVALID_PARTITION (p. 1628)	Invalid partition
PRL_INVALID_VLAN_TAG (p. 1628)	constant for invalid VLAN tag
PRL_IO_GAMMARAMP_DEFAULT (p. 1628)	
PRL_IO_GAMMARAMP_RGB256x3x16 (p. 1628)	
PRL_IO_MAX_DISPLAYS (p. 1628)	
PRL_IO_MAX_ENCODINGS (p. 1629)	
PRL_LICENSE_IS_EXPIRED (p. 1629)	
PRL_LICENSE_IS_VALID (p. 1629)	Success license state
PRL_LINUX_EXTENDED_PARTITION (p. 1629)	
PRL_LINUX_PARTITION_DATA (p. 1630)	
PRL_LINUX_PARTITION_SWAP (p. 1630)	Linux partitions
PRL_MACOS_BOOT_PARTITION (p. 1630)	
PRL_MACOS_PARTITION (p. 1630)	Mac OS
PRL_MAX_VLAN_TAG (p. 1630)	constant for maximum valid VLAN tag (4095)
PRL_METHOD (p. 1631)	Default method decoration
PRL_METHOD_PTR (p. 1631)	
PRL_MSDOS_EXTENDED_PARTITION (p. 1631)	• Disk partition types Extended partitions
PRL_NULL (p. 1631)	NULL (invalid) handle
PRL_PRIVILEGED_GUEST_OS_SESSION (p. 1632)	Guest OS sessions types and constants declarations
PRL_PROXY_MAX_HOSTS (p. 1632)	
PRL_RESULT_DECLARE_ERROR (p. 1632)	Declaration of the error value goes here
PRL_RESULT_DECLARE_SUCCESS (p. 1632)	
PRL_SATA_READ_REQUESTS_PTRN (p. 1633)	Total count of read requests to SATA controller
PRL_SATA_READ_TOTAL_PTRN (p. 1633)	Total count of read bytes through SATA controller
PRL_SATA_WRITE_REQUESTS_PTRN (p. 1633)	Total count of write requests to SATA controller
PRL_SATA_WRITE_TOTAL_PTRN (p. 1633)	Total count of write bytes through SATA controller
PRL_SCSI_READ_REQUESTS_PTRN (p. 1634)	Total count of read requests to SCSI controller
PRL_SCSI_READ_TOTAL_PTRN (p. 1634)	Total count of read bytes through SCSI controller
PRL_SCSI_WRITE_REQUESTS_PTRN (p. 1634)	Total count of write requests to SCSI controller
PRL_SCSI_WRITE_TOTAL_PTRN (p. 1634)	Total count of write bytes through SCSI controller
PRL_SDK_WRAP_FOR_EACH (p. 1635)	
PRL_STRUCT (p. 1651)	

PRL_SUCCEEDED (p. 1651)	Success and failure result codes
PRL_TERNARY_FALSE (p. 1652)	
PRL_TERNARY_TRUE (p. 1652)	
PRL_TERNARY_UNDEF (p. 1652)	
PRL_TRUE (p. 1652)	
PRL_VM_STOP_MODE_MASK (p. 1652)	
PRL_WIN_EXTENDED_PARTITION (p. 1653)	
PRL_WINDOWS_PARTITION_FAT (p. 1653)	
PRL_WINDOWS_PARTITION_FAT32 (p. 1653)	
PRL_WINDOWS_PARTITION_FAT32_LBA (p. 1653)	
PRL_WINDOWS_PARTITION_NTFS (p. 1653)	Windows partitions
PRL_WNG_NO_OPERATION_SYSTEM_INSTALLED (p. 1654)	
PRL_WNG_START_VM_ON_MOUNTED_DISK (p. 1654)	

PRL_ASYNC_METHOD Macro

Syntax

```
#define PRL_ASYNC_METHOD( func ) PRL_EXTERN __declspec(dllexport) PRL_HANDLE PRL_CALL func
```

File

PrlTypes.h

PRL_ASYNC_METHOD_PTR Macro

Syntax

```
#define PRL_ASYNC_METHOD_PTR( func_type ) PRL_HANDLE ( PRL_CALL * func_type )
```

File

PrlTypes.h

PRL_BSD_PARTITION Macro

Free BSD partition.

Syntax

```
#define PRL_BSD_PARTITION 0xA5
```

File

PrlDisk.h

PRL_CALL Macro

Default calling conversion for every supported platform.

Syntax

```
#define PRL_CALL __cdecl
```

File

PrlTypes.h

PRL_CHECKED_DISK_INVALID Macro

Syntax

```
#define PRL_CHECKED_DISK_INVALID PRL_RESULT_DECLARE_SUCCESS(25002)
```

File

PrlErrorsValues.h

PRL_CHECKED_DISK_OLD_VERSION Macro

Syntax

```
#define PRL_CHECKED_DISK_OLD_VERSION PRL_RESULT_DECLARE_SUCCESS(25001)
```

File

PrlErrorsValues.h

PRL_CHECKED_DISK_VALID Macro

- Disk validation errors

+++ ALIGNED VALUE TO 25000

Syntax

```
#define PRL_CHECKED_DISK_VALID PRL_RESULT_DECLARE_SUCCESS(25000)
```

File

PrlErrorsValues.h

PRL_CURRENT_GUEST_OS_SESSION Macro

Syntax

```
#define PRL_CURRENT_GUEST_OS_SESSION "4a5533a7-31c6-4d7a-a400-1f330dc57a9d"
```

File

PrlEnums.h

PRL_DATA_PARTITION Macro

Special handle for data partition from GPT (can be linux, windows, any type).

Syntax

```
#define PRL_DATA_PARTITION 0xd8
```

File

PrlDisk.h

PRL_DISK_NO_IMAGE_CREATION Macro

So not create image when creating disk, just put reference to file in XML

Syntax

```
#define PRL_DISK_NO_IMAGE_CREATION 0x00001000
```

File

PrlDisk.h

PRL_DISPATCHER_LISTEN_PORT Macro

Syntax

```
#define PRL_DISPATCHER_LISTEN_PORT 64000
```

File

PrlPorts.h

PRL_DISPATCHER_LISTEN_PORT_PMOBILE Macro

Syntax

```
#define PRL_DISPATCHER_LISTEN_PORT_PMOBILE 62319
```

File

PrlPorts.h

PRL_EFI_PARTITION Macro

Syntax

```
#define PRL_EFI_PARTITION 0xEF
```

File

PrlDisk.h

PRL_EFI_PARTITION_GPT Macro

EFI system partition

Syntax

```
#define PRL_EFI_PARTITION_GPT 0xEE
```

File

PrlDisk.h

PRL_EMPTY_PARTITION Macro

Syntax

```
#define PRL_EMPTY_PARTITION 0x00
```

File

PrlDisk.h

PRL_EXTERN Macro

Default extern "C" if cplusplus is used

Syntax

```
#define PRL_EXTERN extern "C"
```

File

PrlTypes.h

PRL_FAILED Macro

Syntax

```
#define PRL_FAILED( err ) ( ( PRL_RESULT )( err ) < 0 )
```

File

PrlErrors.h

PRL_FALSE Macro

Syntax

```
#define PRL_FALSE (0)
```

File

PrlTypes.h

PRL_HOST_CPU_TIME_PTRN Macro

Host time change since the last query, in microseconds

Syntax

```
#define PRL_HOST_CPU_TIME_PTRN "host.cpu.time"
```

File

PrlPerfCounters.h

PRL_IDE_READ_REQUESTS_PTRN Macro

Total count of read requests to IDE controller

Syntax

```
#define PRL_IDE_READ_REQUESTS_PTRN "devices.ide#.read_requests"
```

File

PrlPerfCounters.h

PRL_IDE_READ_TOTAL_PTRN Macro

Total count of read bytes through IDE controller

Syntax

```
#define PRL_IDE_READ_TOTAL_PTRN "devices.ide#.read_total"
```

File

PrlPerfCounters.h

PRL_IDE_WRITE_REQUESTS_PTRN Macro

Total count of write requests to IDE controller

Syntax

```
#define PRL_IDE_WRITE_REQUESTS_PTRN "devices.ide#.write_requests"
```

File

PrlPerfCounters.h

PRL_IDE_WRITE_TOTAL_PTRN Macro

Total count of write bytes through IDE controller

Syntax

```
#define PRL_IDE_WRITE_TOTAL_PTRN "devices.ide#.write_total"
```

File

PrlPerfCounters.h

PRL_INFO_VM_MIGRATE_STORAGE_IS_SHARED Macro

Syntax

```
#define PRL_INFO_VM_MIGRATE_STORAGE_IS_SHARED PRL_RESULT_DECLARE_SUCCESS(31028)
```

File

PrlErrorsValues.h

PRL_INVALID_HANDLE Macro

Syntax

```
#define PRL_INVALID_HANDLE ( 0 )
```

File

PrlTypes.h

PRL_INVALID_PARTITION Macro

Invalid partition

Syntax

```
#define PRL_INVALID_PARTITION 0xFF
```

File

PrlDisk.h

PRL_INVALID_VLAN_TAG Macro

constant for invalid VLAN tag

Syntax

```
#define PRL_INVALID_VLAN_TAG 0xffff
```

File

PrlEnums.h

PRL_IO_GAMMARAMP_DEFAULT Macro

Syntax

```
#define PRL_IO_GAMMARAMP_DEFAULT 1
```

File

PrlIOStructs.h

PRL_IO_GAMMARAMP_RGB256x3x16 Macro

Syntax

```
#define PRL_IO_GAMMARAMP_RGB256x3x16 2
```

File

PrlIOStructs.h

PRL_IO_MAX_DISPLAYS Macro

Syntax

```
#define PRL_IO_MAX_DISPLAYS 16
```

File

PrlIOStructs.h

PRL_IO_MAX_ENCODINGS Macro

Syntax

```
#define PRL_IO_MAX_ENCODINGS 32
```

File

PrlIOStructs.h

PRL_LICENSE_IS_EXPIRED Macro

Syntax

```
#define PRL_LICENSE_IS_EXPIRED( x ) \
    ( ( x == PRL_ERR_LICENSE_EXPIRED ) || \
        ( x == PRL_ERR_VZLICENSE_KA_LICENSE_EXPIRED ) )
```

File

PrlErrors.h

PRL_LICENSE_IS_VALID Macro

Success license state

Syntax

```
#define PRL_LICENSE_IS_VALID(x) \
    ( (PRL_RESULT) ( ( x == PRL_ERR_LICENSE_VALID ) || \
        ( x == PRL_ERR_LICENSE_GRACED ) ) )
```

File

PrlErrors.h

PRL_LINUX_EXTENDED_PARTITION Macro

Syntax

```
#define PRL_LINUX_EXTENDED_PARTITION 0x85
```

File

PrlDisk.h

PRL_LINUX_PARTITION_DATA Macro

Syntax

```
#define PRL_LINUX_PARTITION_DATA 0x83
```

File

PrlDisk.h

PRL_LINUX_PARTITION_SWAP Macro

Linux partitions

Syntax

```
#define PRL_LINUX_PARTITION_SWAP 0x82
```

File

PrlDisk.h

PRL_MACOS_BOOT_PARTITION Macro

Syntax

```
#define PRL_MACOS_BOOT_PARTITION 0xAB
```

File

PrlDisk.h

PRL_MACOS_PARTITION Macro

Mac OS

Syntax

```
#define PRL_MACOS_PARTITION 0xAF
```

File

PrlDisk.h

PRL_MAX_VLAN_TAG Macro

constant for maximum valid VLAN tag (4095)

Syntax

```
#define PRL_MAX_VLAN_TAG 0xfff
```

File

PrlEnums.h

PRL_METHOD Macro

Default method decoration

Syntax

```
#define PRL_METHOD( func ) PRL_EXTERN __declspec(dllexport) PRL_RESULT PRL_CALL func
```

File

PrlTypes.h

PRL_METHOD_PTR Macro

Syntax

```
#define PRL_METHOD_PTR( func_type ) PRL_RESULT ( PRL_CALL * func_type )
```

File

PrlTypes.h

PRL_MSDOS_EXTENDED_PARTITION Macro

- Disk partition types

Extended partitions

Syntax

```
#define PRL_MSDOS_EXTENDED_PARTITION 0x05
```

File

PrlDisk.h

PRL_NULL Macro

NULL (invalid) handle

Syntax

```
#define PRL_NULL ( (PRL_VOID_PTR) 0 )
```

File

PrlTypes.h

PRL_PRIVILEGED_GUEST_OS_SESSION Macro

Guest OS sessions types and constants declarations

Syntax

```
#define PRL_PRIVILEGED_GUEST_OS_SESSION "531582ac-3dce-446f-8c26-dd7e3384dcf4"
```

File

PrlEnums.h

PRL_PROXY_MAX_HOSTS Macro

Syntax

```
#define PRL_PROXY_MAX_HOSTS 16
```

File

PrlIOStructs.h

PRL_RESULT_DECLARE_ERROR Macro

Declaration of the error value goes here

Syntax

```
#define PRL_RESULT_DECLARE_ERROR( err ) ( (PRL_RESULT) (0x80000000 + 0x##err) )
```

File

PrlErrors.h

PRL_RESULT_DECLARE_SUCCESS Macro

Syntax

```
#define PRL_RESULT_DECLARE_SUCCESS( err ) ( (PRL_RESULT) (err) )
```

File

PrlErrors.h

PRL_SATA_READ_REQUESTS_PTRN Macro

Total count of read requests to SATA controller

Syntax

```
#define PRL_SATA_READ_REQUESTS_PTRN "devices.sata#.read_requests"
```

File

PrlPerfCounters.h

PRL_SATA_READ_TOTAL_PTRN Macro

Total count of read bytes through SATA controller

Syntax

```
#define PRL_SATA_READ_TOTAL_PTRN "devices.sata#.read_total"
```

File

PrlPerfCounters.h

PRL_SATA_WRITE_REQUESTS_PTRN Macro

Total count of write requests to SATA controller

Syntax

```
#define PRL_SATA_WRITE_REQUESTS_PTRN "devices.sata#.write_requests"
```

File

PrlPerfCounters.h

PRL_SATA_WRITE_TOTAL_PTRN Macro

Total count of write bytes through SATA controller

Syntax

```
#define PRL_SATA_WRITE_TOTAL_PTRN "devices.sata#.write_total"
```

File

PrlPerfCounters.h

PRL_SCSI_READ_REQUESTS_PTRN Macro

Total count of read requests to SCSI controller

Syntax

```
#define PRL_SCSI_READ_REQUESTS_PTRN "devices.scsi#.read_requests"
```

File

PrlPerfCounters.h

PRL_SCSI_READ_TOTAL_PTRN Macro

Total count of read bytes through SCSI controller

Syntax

```
#define PRL_SCSI_READ_TOTAL_PTRN "devices.scsi#.read_total"
```

File

PrlPerfCounters.h

PRL_SCSI_WRITE_REQUESTS_PTRN Macro

Total count of write requests to SCSI controller

Syntax

```
#define PRL_SCSI_WRITE_REQUESTS_PTRN "devices.scsi#.write_requests"
```

File

PrlPerfCounters.h

PRL_SCSI_WRITE_TOTAL_PTRN Macro

Total count of write bytes through SCSI controller

Syntax

```
#define PRL_SCSI_WRITE_TOTAL_PTRN "devices.scsi#.write_total"
```

File

PrlPerfCounters.h

PRL_SDK_WRAP_FOR_EACH Macro

Syntax

```
#define PRL_SDK_WRAP_FOR_EACH \
PRL_SDK_WRAP_FOR_EACH_DEPRECATED() \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_Init ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_InitEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_Deinit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetAppMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetResultDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetMessageType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_MsgCanBeIgnored ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetCrashDumpsPath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_InitCrashHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBuffer_Free ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_CreateStringsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_CreateHandlesList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_CreateOpTypeList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetSupportedOsesTypes ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetSupportedOsesVersions ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetDefaultOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_SendProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_SendPackedProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStrList_GetItemsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStrList_AddItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStrList_GetItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStrList_RemoveItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHndlList_GetItemsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHndlList_AddItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHndlList_GetItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHndlList_RemoveItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOpTypeList_GetItemsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOpTypeList_AddItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOpTypeList_GetItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOpTypeList_RemoveItem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOpTypeList_GetTypeSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GetRecommendMinVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_AddRef ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_Free ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDbg_GetHandlesNum ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDbg_PrlResultToString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDbg_HandleTypeToString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDbg_EventTypeToString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_Wait ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_Cancel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetStatus ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetProgress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetRetCode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetDataPtr ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetResult ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetError ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_GetOpCode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlJob_IsRequestWasSent ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlResult_GetParamsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlResult_GetParamByIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlResult_GetParam ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlResult_GetParamByIndexAsString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlResult_GetParamAsString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetType ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetDataPtr ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetServer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetJob ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetParamsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetParam ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetParamByName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetErrCode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetErrString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_CanBeIgnored ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_IsAnswerRequired ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetIssuerType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_GetIssuerId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_CreateAnswerEvent ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToCData ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToUint32 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToInt32 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToUint64 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToInt64 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToBoolean ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_ToHandle ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvtPrm_GetBuffer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_Login ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_LoginLocal ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_Logoff ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RegEventHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UnregEventHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetQuestions ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_SetNonInteractiveSession ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_IsNonInteractiveSession ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_DisableConfirmationMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_EnableConfirmationMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_IsConfirmationModeEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetSrvConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetCommonPrefs ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CommonPrefsBeginEdit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CommonPrefsCommit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CommonPrefsCommitEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetUserProfile ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetUserInfoList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetUserInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetVirtualNetworkList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_AddVirtualNetwork ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateVirtualNetwork ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_DeleteVirtualNetwork ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_SetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_GetPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_SetPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_IsUsedByDefault ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOffmgmtService_SetUsedByDefault ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateOfflineService ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_DeleteOfflineService ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetOfflineServicesList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkClass_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkClass_GetClassId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkClass_SetClassId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkClass_GetNetworkList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkClass_SetNetworkList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateNetworkClassesList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetNetworkClassesList ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_GetClassId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_SetClassId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_GetTotalRate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_SetTotalRate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_SetDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_GetDevice) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_SetRate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_GetRate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_SetRateMPU ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingEntry_GetRateMPU ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateNetworkShapingConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetNetworkShapingConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingConfig_IsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingConfig_SetEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingConfig_GetNetworkShapingList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingConfig_SetNetworkShapingList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingBandwidthEntry_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingBandwidthEntry_SetDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingBandwidthEntry_GetDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingBandwidthEntry_SetBandwidth ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingBandwidthEntry_GetBandwidth) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingConfig_GetNetworkDeviceBandwidthList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkShapingConfig_SetNetworkDeviceBandwidthList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkRate_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkRate_GetClassId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetworkRate_GetRate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetNetworkRateList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetNetworkRateList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsRateBound ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetRateBound ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOsTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOsTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_ApplyConfigSample ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAppTemplateList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAppTemplateList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_ConfigureGenericPci ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetStatistics ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UserProfileBeginEdit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UserProfileCommit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_IsConnected ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetServerInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RegisterVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RegisterVmEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RegisterVmWithUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_Register3rdPartyVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CreateVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetVmList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetVmListEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetDefaultVmConfig) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CreateVmBackup ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RestoreVmBackup ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetBackupTree ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RemoveVmBackup ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_SubscribeToHostStatistics ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UnsubscribeFromHostStatistics ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_Shutdown ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_ShutdownEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsGetDiskList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsGetDirEntries ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsCreateDir ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsRemoveEntry ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsCanCreateFile ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsRenameEntry ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateLicense ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateLicenseEx ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetLicenseInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_SendAnswer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_StartSearchVms ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsInfo_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsInfo_GetFsType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsInfo_GetChildEntriesCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsInfo_GetChildEntry ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsInfo_GetParentEntry ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsEntry_GetAbsolutePath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsEntry_GetRelativeName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsEntry_GetLastModifiedDate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsEntry_GetSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsEntry_GetPermissions ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFsEntry_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_NetServiceStart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_NetServiceStop ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_NetServiceRestart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_NetServiceRestoreDefaults ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetNetServiceStatus ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetPackedProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_AttachToLostTask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetSupportedOses ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuModel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuSpeed ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuHvt ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuFeaturesEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetCpuFeaturesMaskingCapabilities ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostOsType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostOsMajor ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostOsMinor ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostOsSubMinor ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostOsStrPresentation ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_IsSoundDefaultEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_IsUsbSupported ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_IsVtdSupported ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetMaxHostNetAdapters ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetMaxVmNetAdapters ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHostname ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetDefaultGateway ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetDefaultGatewayIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetDnsServers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSearchDomains ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetFloppyDisksCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetFloppyDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetOpticalDisksCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetOpticalDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSerialPortsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSerialPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetParallelPortsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetParallelPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSoundOutputDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSoundOutputDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSoundMixerDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetSoundMixerDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetPrintersCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetPrinter ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetGenericPciDevicesCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetGenericPciDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetGenericScsiDevicesCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetGenericScsiDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetUsbDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetUsbDev ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHardDisksCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetHardDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetNetAdaptersCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfg_GetNetAdapter ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgDev_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgDev_GetId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgDev_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgDev_IsConnectedToVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgDev_GetDeviceState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgDev_SetDeviceState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHdd_GetDevName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHdd_GetDevId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHdd_GetDevSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHdd_GetDiskIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHdd_GetPartsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHdd_GetPart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_GetSysName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_GetSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_GetIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_IsInUse ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_IsLogical ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgHddPart_IsActive ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetNetAdapterType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetSysIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_IsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_IsConfigureWithDhcp ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_IsConfigureWithDhcpIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetDefaultGateway ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetDefaultGatewayIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetMacAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetVlanTag ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetNetAddresses ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetDnsServers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgNet_GetSearchDomains ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgPci_GetDeviceClass ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvCfgPci_IsPrimaryDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_GetVmDirUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_SetVmDirUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_GetDefaultVmFolder ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_SetDefaultVmFolder ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_CanUseMngConsole ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_CanChangeSrvSets ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrCfg_IsLocalAdministrator ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrInfo_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrInfo_GetUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrInfo_GetSessionCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrInfo_GetDefaultVmFolder ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsrInfo_CanChangeSrvSets ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetDefaultVmDir ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetDefaultCtDir ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetReservedMemLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetReservedMemLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetMinVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetMinVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetMaxVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetMaxVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetRecommendMaxVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetRecommendMaxVmMem ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetMaxReservMemLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetMaxReservMemLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetMinReservMemLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetMinReservMemLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_IsAdjustMemAuto ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetAdjustMemAuto ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_IsSendStatisticReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetSendStatisticReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetDefaultVNCHostName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetDefaultVNCHostName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetVNCBasePort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetVNCBasePort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_CanChangeDefaultSettings ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetCanChangeDefaultSettings ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetMinSecurityLevel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetMinSecurityLevel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetConfirmationsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetConfirmationsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetDefaultBackupServer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetDefaultBackupServer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetBackupUserLogin ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetBackupUserLogin ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetBackupUserPassword ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_IsBackupUserPasswordEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetBackupUserPasswordEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetDefaultBackupDirectory ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetDefaultBackupDirectory ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetBackupTimeout ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetBackupTimeout ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_ArePluginsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_EnablePlugins ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetVmCpuLimitType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetVmCpuLimitType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_IsVerboseLogEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetVerboseLogEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_IsAllowMultiplePMC ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetAllowMultiplePMC ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_IsLogRotationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetLogRotationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetCpuFeaturesMaskEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetCpuFeaturesMaskEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCpuFeatures_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCpuFeatures_GetValue ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCpuFeatures_SetValue ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetNetworkId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetNetworkId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetNetworkType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetNetworkType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetBoundCardMac ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetBoundCardMac ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetAdapterName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetAdapterName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetAdapterIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetAdapterIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetHostIPAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetHostIPAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetHostIP6Address ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetHostIP6Address ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetDhcpIPAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetDhcpIPAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetDhcpIP6Address ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetDhcpIP6Address ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetIPNetMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetIPNetMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetIP6NetMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetIP6NetMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetVlanTag ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetVlanTag ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetIPScopeStart ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetIPScopeStart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetIPScopeEnd ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetIPScopeEnd ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetIP6ScopeStart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetIP6ScopeStart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetIP6ScopeEnd ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetIP6ScopeEnd ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_IsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_IsAdapterEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetAdapterEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_IsDHCPServerEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetDHCPServerEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_IsDHCP6ServerEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetDHCP6ServerEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_IsNATServerEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetNATServerEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetPortForwardList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_SetPortForwardList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVirtNet_GetBoundAdapterInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_GetRuleName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_SetRuleName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_GetIncomingPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_SetIncomingPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_GetRedirectIPAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_SetRedirectIPAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_GetRedirectPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_SetRedirectPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_GetRedirectVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPortFwd_SetRedirectVm ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Start ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_StartEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Restart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Stop ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_StopEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Pause ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Reset ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Suspend ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_ChangeSid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_ResetUptime ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetSuspendedScreen ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Resume ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_DropSuspendedState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CreateSnapshot ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_SwitchToSnapshot ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_SwitchToSnapshotEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_DeleteSnapshot ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetSnapshotsTree ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetSnapshotsTreeEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Lock ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Unlock ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_UpdateSnapshotData ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Clone ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CloneEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CloneWithUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Migrate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_MigrateEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_MigrateWithRename ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_MigrateWithRenameEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_MigrateCancel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GenerateVmDevFilename ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Delete ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetPackedProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetState ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_RefreshConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_RefreshConfigEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_LoginInGuest ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_StartVncServer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_StopVncServer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmGuest_Logout ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmGuest_RunProgram ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmGuest_GetNetworkSettings ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_SetUserPasswd ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_SetConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetDefaultConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsConfigInvalid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetConfigValidity ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_AddDefaultDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_AddDefaultDeviceEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsDefaultDeviceNeeded ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDefaultMemSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDefaultHddSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDefaultVideoRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetStatistics ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetStatisticsEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_SubscribeToGuestStatistics ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_UnsubscribeFromGuestStatistics ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Reg ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_RegEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Unreg ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Restore ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_BeginEdit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Commit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CommitEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_CreateVmDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAccessRights ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAllDevices ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDevsCountByType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDevByType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetFloppyDisksCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetFloppyDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetHardDisksCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetHardDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOpticalDisksCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOpticalDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetParallelPortsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetParallelPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSerialPortsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSerialPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSoundDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSoundDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetUsbDevicesCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetUsbDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetNetAdaptersCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetNetAdapter ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetGenericPciDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetGenericPciDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetGenericScsiDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetGenericScsiDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDisplayDevsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDisplayDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_CreateShare ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSharesCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetShare ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSmartGuardEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartGuardEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSmartGuardNotifyBeforeCreation ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartGuardNotifyBeforeCreation ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSmartGuardInterval ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartGuardInterval ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSmartGuardMaxSnapshotsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartGuardMaxSnapshotsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUserDefinedSharedFoldersEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUserDefinedSharedFoldersEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSmartMountEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartMountEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSmartMountRemovableDrivesEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartMountRemovableDrivesEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSmartMountDVDsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartMountDVDsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSmartMountNetworkSharesEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSmartMountNetworkSharesEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSharedProfileEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSharedProfileEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUseDesktop ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUseDesktop ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUseDocuments ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUseDocuments ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUsePictures ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUsePictures ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUseMusic ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUseMusic ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUseDownloads ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUseDownloads ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUseMovies ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUseMovies ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOptimizeModifiersMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOptimizeModifiersMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsShareClipboard ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetShareClipboard ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsOfflineManagementEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOfflineManagementEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsToolsAutoUpdateEnabled) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetToolsAutoUpdateEnabled) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsTimeSynchronizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetTimeSynchronizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsTimeSyncSmartModeEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetTimeSyncSmartModeEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetTimeSyncInterval ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetTimeSyncInterval ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsAllowSelectBootDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAllowSelectBootDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_RegEventHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_UnregEventHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetQuestions ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_CreateBootDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetBootDevCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetBootDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetHostname ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetHostname ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDnsServers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetDnsServers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCtId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetLinkedVmUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOsType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVideoRamSize ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVideoRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetMemGuaranteeSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetMemGuaranteeSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuAccelLevel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuAccelLevel ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsCpuVtxEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsCpuHotplugEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuHotplugEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_Get3DAccelerationMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_Set3DAccelerationMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsVerticalSynchronizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVerticalSynchronizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsHighResolutionEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetHighResolutionEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsAdaptiveHypervisorEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAdaptiveHypervisorEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsSwitchOffWindowsLogoEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSwitchOffWindowsLogoEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsLongerBatteryLifeEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetLongerBatteryLifeEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsBatteryStatusEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetBatteryStatusEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsNestedVirtualizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetNestedVirtualizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsPMUVirtualizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetPMUVirtualizationEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsLockGuestOnSuspendEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetLockGuestOnSuspendEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsIsolatedVmEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetIsolatedVmEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuUnits ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuUnits ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuLimitEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuLimitEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCpuMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCpuMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetNodeMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetNodeMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetIoPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetIoPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetIoLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetIoLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetIopsLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetIopsLimit ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetServerUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetServerHost ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetHomePath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetLocation ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetIcon ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetIcon ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetTemplateSign ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCustomProperty ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCustomProperty ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAutoStart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAutoStart ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAutoStartDelay ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAutoStartDelay ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetStartLoginMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetStartLoginMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetStartUserLogin ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetStartUserCreds ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAutoStop ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAutoStop ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetActionOnWindowClose ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetActionOnWindowClose ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetActionOnStopMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetActionOnStopMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetWindowMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetWindowMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsStartInDetachedWindowEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetStartInDetachedWindowEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsLockInFullScreenMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetLockInFullScreenMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetLastModifiedDate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetLastModifierName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetUptimeStartDate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetUptime ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsGuestSharingEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetGuestSharingEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsGuestSharingAutoMount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetGuestSharingAutoMount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsGuestSharingEnableSpotlight ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetGuestSharingEnableSpotlight ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsHostSharingEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetHostSharingEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsShareAllHostDisks ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetShareAllHostDisks ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsShareUserHomeDir ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetShareUserHomeDir ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsMapSharedFoldersOnLetters ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetMapSharedFoldersOnLetters ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVNCMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVNCMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVNCPassword ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVNCPassword ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVNCHostName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVNCHostName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVNCPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVNCPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVNCWebSocketPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsVNCEncryptionEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsScrResEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetScrResEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsDiskCacheWriteBack ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetDiskCacheWriteBack ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsOsResInFullScrMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOsResInFullScrMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsCloseAppOnShutdown ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCloseAppOnShutdown ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSystemFlags ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSystemFlags ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsDisableAPIC ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetDisableAPICSign ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsDisableSpeaker ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetDisableSpeakerSign ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetUndoDisksMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUndoDisksMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetForegroundPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetForegroundPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetBackgroundPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetBackgroundPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsUseDefaultAnswers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetUseDefaultAnswers ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetSearchDomains ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetSearchDomains ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOfflineServices ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOfflineServices ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetConfirmationsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetConfirmationsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsAutoCompressEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAutoCompressEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetAutoCompressInterval ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAutoCompressInterval ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetFreeDiskSpaceRatio ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetFreeDiskSpaceRatio ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVmInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetVmType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetVmType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsEfiEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetEfiEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetExternalBootDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetExternalBootDevice ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CreateEvent ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CreateUnattendedFloppy ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CreateUnattendedCd ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_InitiateDevStateNotifications ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_ValidateConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_UpdateSecurity ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_InstallTools ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetToolsState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_Connect ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_Disconnect ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_CreateImage ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_CopyImage ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_ResizeImage ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_Remove ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_IsConnected ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetConnected ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_IsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_IsRemote ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetRemote ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetEmulatedType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetEmulatedType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetImagePath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetImagePath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetSysName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetSysName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetFriendlyName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetFriendlyName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetIfaceType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetIfaceType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetSubType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetSubType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetStackIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetStackIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetDefaultStackIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_GetOutputFile ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetOutputFile ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_IsPassthrough ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDev_SetPassthrough ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetDiskType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetDiskType ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_IsSplitted ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetSplitted ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetDiskSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetDiskSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetSizeOnDisk ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_AddPartition ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetPartitionsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetPartition ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHdPart_Remove ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetMountPoint ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetMountPoint ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetAutoCompressEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_IsAutoCompressEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetStorageURL ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetStorageURL ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetEncryption ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetEncryption ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_GetSerialNumber ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHd_SetSerialNumber ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHdPart_GetSysName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHdPart_SetSysName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHdEncryption_GetKeyId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevHdEncryption_SetKeyId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetBoundAdapterIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetBoundAdapterIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetBoundAdapterName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetBoundAdapterName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetMacAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetMacAddressCanonical ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetMacAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GenerateMacAddr ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsAutoApply ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetAutoApply ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetNetAddresses ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetNetAddresses ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetDnsServers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetDnsServers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetSearchDomains ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetSearchDomains ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsConfigureWithDhcp ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetConfigureWithDhcp ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsConfigureWithDhcpIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetConfigureWithDhcpIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetDefaultGateway ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetDefaultGateway ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetDefaultGatewayIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetDefaultGatewayIPv6 ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetVirtualNetworkId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetVirtualNetworkId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetAdapterType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetAdapterType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsPktFilterPreventMacSpoof ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetPktFilterPreventMacSpoof) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsPktFilterPreventPromisc ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetPktFilterPreventPromisc ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsPktFilterPreventIpSpoof ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetPktFilterPreventIpSpoof ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_IsFirewallEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetFirewallEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetFirewallDefaultPolicy ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetFirewallDefaultPolicy ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetFirewallRuleList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetFirewallRuleList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_GetHostInterfaceName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevNet_SetHostInterfaceName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_Create ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_GetLocalPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_SetLocalPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_GetRemotePort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_SetRemotePort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_GetProtocol ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_SetProtocol ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_GetLocalNetAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_SetLocalNetAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_GetRemoteNetAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFirewallRule_SetRemoteNetAddress ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevUsb_GetAutoconnectOption ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevUsb_SetAutoconnectOption ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevSound_GetOutputDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevSound_SetOutputDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevSound_GetMixerDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevSound_SetMixerDev ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevSerial_GetSocketMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmDevSerial_SetSocketMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_Remove ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_SetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_GetPath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_SetPath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_GetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_SetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_IsEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_SetEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_IsReadOnly ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlShare_SetReadOnly ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_Remove ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_SetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_GetIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_SetIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_GetSequenceIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_SetSequenceIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_IsInUse ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBootDev_SetInUse ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmInfo_GetState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmInfo_GetAccessRights ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmInfo_IsInvalid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmInfo_IsVmWaitingForAnswer ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmInfo_IsVncServerStarted ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFoundVmInfo_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFoundVmInfo_IsOldConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFoundVmInfo_GetOSVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFoundVmInfo_GetConfigPath ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlFoundVmInfo_IsTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlAcl_IsAllowed ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlAcl_GetAccessForOthers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlAcl_SetAccessForOthers ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlAcl_GetOwnerName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlAcl_IsCurrentSessionOwner ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmToolsInfo_GetState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmToolsInfo_GetVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetTotalRamSize )    \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetUsageRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetFreeRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetRealRamSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetOsUptime ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetCpusStatsCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStat_GetCpuStat ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStatCpu_GetCpuUsage ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStatCpu_GetTotalTime ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStatCpu_GetUserTime ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlStatCpu_GetSystemTime ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLic_IsValid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLic_GetStatus ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLic_GetLicenseKey ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLic_GetUserName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLic_GetCompanyName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetHostName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetProductVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetCmdPort ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetApplicationMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetServerUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetStartTime ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrvInfo_GetStartTimeMonotonic ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlNetSvc_GetStatus ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_FsGenerateEntryName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLoginResponse_GetSessionUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLoginResponse_GetRunningTaskCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLoginResponse_GetRunningTaskByIndex ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLoginResponse_GetServerUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLoginResponse_GetHostOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlLoginResponse_GetProductVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlRunningTask_GetTaskUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlRunningTask_GetTaskType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlRunningTask_GetTaskParametersAsString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_SubscribeToPerfStats )     \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UnsubscribeFromPerfStats ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetPerfStats )             \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_SubscribeToPerfStats )      \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_UnsubscribeFromPerfStats )  \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_GetPerfStats ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_AuthWithGuestSecurityDb ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmInfo_GetAdditionState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Compact ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CancelCompact ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_ConvertDisks ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CancelConvertDisks ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOsesMatrix_GetSupportedOsesTypes ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOsesMatrix_GetSupportedOsesVersions ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOsesMatrix_GetDefaultOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlOsesMatrix_GetSupportType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_HasRestriction ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetRestrictionInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_CreateProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_SetUserName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetUserName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_SetUserEmail ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetUserEmail ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_SetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_SetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_SetReason ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetReason ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetScheme ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetArchiveFileName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_GetData ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_AsString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_Assembly ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlReport_Send ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_InstallAppliance ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CancelInstallAppliance ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_StopInstallAppliance ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlAppliance_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetDescription ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetOsType ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetOsVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetOsTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_GetCpuMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCtTemplate_IsCached ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetCtTemplateList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RemoveCtTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetUsbIdentityCount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetUsbIdentity ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsbIdent_GetSystemName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsbIdent_GetFriendlyName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlUsbIdent_GetVmUuidAssociation ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_SetUsbIdentAssociation ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_CopyCtTemplate ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_IsFeatureSupported ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_AddIPPrivateNetwork ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RemoveIPPrivateNetwork ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_UpdateIPPrivateNetwork ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetIPPrivateNetworksList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_Create ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_SetNetAddresses ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_GetNetAddresses ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_SetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_IsGlobal ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlIPPrivNet_SetGlobal ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsRamHotplugEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetRamHotplugEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RefreshPlugins ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetPluginsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Mount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Umount ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPluginInfo_GetVendor ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPluginInfo_GetCopyright ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPluginInfo_GetShortDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPluginInfo_GetLongDescription ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPluginInfo_GetVersion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlPluginInfo_GetId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_LoginEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_LoginLocalEx ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_RegEventHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_UnregEventHandler ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetDiskFreeSpace ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlBackupResult_GetBackupUuid ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Move ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Connect ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_Disconnect ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetVmConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetResource ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetResource ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsAutoApplyIpOnly) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetAutoApplyIpOnly) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetCapabilitiesMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetCapabilitiesMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetNetfilterMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetNetfilterMode ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_GetPackageId ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetCPUPoolsList ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_MoveToCPUPool ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_RecalculateCPUPool ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDispCfg_GetCpuPool ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCPUPool_GetVendor ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCPUPool_GetName ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCPUPool_GetCpuFeaturesMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetFeaturesMask ) \
```

```
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetFeaturesMask ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_SendProblemReport ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_FromString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlHandle_ToString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlEvent_FromString ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_SwitchConsoleLogging ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetHighAvailabilityEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_IsHighAvailabilityEnabled ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetHighAvailabilityPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetHighAvailabilityPriority ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_CreateDisk_Local ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_OpenDisk_Local ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_WaitForCompletion ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_SwitchToState ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_Write ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_Read ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDisk_GetDiskInfo ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDiskMap_GetSize ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDiskMap_GetGranularity ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlDiskMap_Read ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CaptureScreen ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVm_CommitEncryption ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetActionOnGuestCrash ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetActionOnGuestCrash ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_GetOptionsOnGuestCrash ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVmCfg_SetOptionsOnGuestCrash ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_GetVcmmdConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlSrv_SetVcmmdConfig ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVcmmdConfig_SetPolicy ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlVcmmdConfig_GetPolicy ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlCt_Reinstall ) \
PRL_SDK_WRAP_FOR_EACH_ITERATOR( PrlApi_GuestToString )
```

File

PrlApiMacro.h

PRL_STRUCT Macro

Syntax

```
#define PRL_STRUCT( struct_name ) __attribute__((packed)) struct_name
```

File

PrlTypes.h

PRL_SUCCEEDED Macro

Success and failure result codes

Syntax

```
#define PRL_SUCCEEDED( err ) ( ( PRL_RESULT )( err ) >= 0 )
```

File

PrlErrors.h

PRL_TERNARY_FALSE Macro

Syntax

```
#define PRL_TERNARY_FALSE (3)
```

File

PrlTypes.h

PRL_TERNARY_TRUE Macro

Syntax

```
#define PRL_TERNARY_TRUE (2)
```

File

PrlTypes.h

PRL_TERNARY_UNDEF Macro

Syntax

```
#define PRL_TERNARY_UNDEF (0)
```

File

PrlTypes.h

PRL_TRUE Macro

Syntax

```
#define PRL_TRUE (1)
```

File

PrlTypes.h

PRL_VM_STOP_MODE_MASK Macro

Syntax

```
#define PRL_VM_STOP_MODE_MASK 0xff
```

File

PrlEnums.h

PRL_WIN_EXTENDED_PARTITION Macro

Syntax

```
#define PRL_WIN_EXTENDED_PARTITION 0x0F
```

File

PrlDisk.h

PRL_WINDOWS_PARTITION_FAT Macro

Syntax

```
#define PRL_WINDOWS_PARTITION_FAT 0x06
```

File

PrlDisk.h

PRL_WINDOWS_PARTITION_FAT32 Macro

Syntax

```
#define PRL_WINDOWS_PARTITION_FAT32 0x0B
```

File

PrlDisk.h

PRL_WINDOWS_PARTITION_FAT32_LBA Macro

Syntax

```
#define PRL_WINDOWS_PARTITION_FAT32_LBA 0x0C
```

File

PrlDisk.h

PRL_WINDOWS_PARTITION_NTFS Macro

Windows partitions

Syntax

```
#define PRL_WINDOWS_PARTITION_NTFS 0x07
```

File

PrlDisk.h

PRL_WNG_NO_OPERATION_SYSTEM_INSTALLED Macro

Syntax

```
#define PRL_WNG_NO_OPERATION_SYSTEM_INSTALLED PRL_RESULT_DECLARE_ERROR(10015)
```

File

PrlErrorsValues.h

PRL_WNG_START_VM_ON_MOUNTED_DISK Macro

Syntax

```
#define PRL_WNG_START_VM_ON_MOUNTED_DISK PRL_RESULT_DECLARE_SUCCESS(439)
```

File

PrlErrorsValues.h

Other Macros

Macros

Name	Description
___PARALLELS_DISK_SDK_DEFINITIONS___ (p. 1656)	
__PARALLELS_API_COMMANDS_FLAGS_H__ (p. 1656)	
__PARALLELS_API_CORE_H__ (p. 1657)	
__PARALLELS_API_CT_H__ (p. 1657)	
__PARALLELS_API_DISPATCHER_H__ (p. 1657)	
__PARALLELS_API_ENUMS_H__ (p. 1657)	
__PARALLELS_API_ERRORS_H__ (p. 1657)	
__PARALLELS_API_ERRORS_VALUES_H__ (p. 1658)	
__PARALLELS_API_EVENTS_VALUES_H__ (p. 1658)	
__PARALLELS_API_FS_H__ (p. 1658)	
__PARALLELS_API_HEADER_H__ (p. 1658)	
__PARALLELS_API_KEYS_H__ (p. 1658)	

__PARALLELS_API_MANAGEMENT_H__ (p. 1659)	
__PARALLELS_API_NET_H__ (p. 1659)	
__PARALLELS_API_OSES_H__ (p. 1659)	
__PARALLELS_API_PORTS_H__ (p. 1659)	
__PARALLELS_API_STATISTIC_H__ (p. 1659)	
__PARALLELS_API_TYPES_H__ (p. 1660)	
__PARALLELS_API_VM_H__ (p. 1660)	
__PARALLELS_IO_STRUCTS_H__ (p. 1660)	
__PARALLELS_PERF_COUNTERS_H__ (p. 1660)	
__PRL_PLUGINS_CLASSES_DEFINED__ (p. 1660)	
__PRLAPIVERSION_H__ (p. 1661)	
CAPDEFAULTMASK (p. 1661)	
FROM_PARALLELS_API_VER (p. 1661)	Converts API version from the library native format to major and minor values.
FS_PERM_EXECUTE (p. 1662)	
FS_PERM_READ (p. 1662)	
FS_PERM_WRITE (p. 1662)	
GUID_CLS_BASE_STR (p. 1662)	
IS_ANDROID (p. 1663)	
IS_CHROMEOS (p. 1663)	
IS_FREEBSD (p. 1663)	
IS_FS_EXECUTE_PERMITTED (p. 1663)	
IS_FS_READ_PERMITTED (p. 1663)	
IS_FS_WRITE_PERMITTED (p. 1664)	
IS_LINUX (p. 1664)	
IS_MACOS (p. 1664)	
IS_MSDOS (p. 1664)	
IS_NETWARE (p. 1664)	
IS_OS2 (p. 1665)	
IS_OTHER (p. 1665)	
IS_SOLARIS (p. 1665)	
IS_VALID_ANDROID_VERSION (p. 1665)	
IS_VALID_CHROMEOS_VERSION (p. 1666)	
IS_VALID_FREEBSD_VERSION (p. 1666)	
IS_VALID_LINUX_VERSION (p. 1666)	
IS_VALID_MACOS_VERSION (p. 1666)	
IS_VALID_MSDOS_VERSION (p. 1667)	
IS_VALID_NETWARE_VERSION (p. 1667)	
IS_VALID_OS2_VERSION (p. 1667)	
IS_VALID_OTHER_OS_VERSION (p. 1667)	
IS_VALID_SOLARIS_VERSION (p. 1668)	
IS_VALID_WINDOWS_VERSION (p. 1668)	

IS_WIN_VER_BELOW (p. 1668)	PVS_GUEST_VER_WIN_OTHER (p. 1618) is used for legacy guest OS (9x and below), but has maximal number 0xFF.
IS_WINDOWS (p. 1668)	
MAX_PARALLEL_COUNT (p. 1669)	
MAX_USB_PRINTER_COUNT (p. 1669)	Now we have 128 usb1.1 devices, but on 2 from it mapped mouse and keyboard
PARALLELS_API_VER (p. 1669)	Current Virtuozzo API version.
PARALLELS_API_VER_1 (p. 1669)	
PARALLELS_API_VER_2 (p. 1670)	
PARALLELS_API_VER_3 (p. 1670)	
PARALLELS_API_VER_4 (p. 1670)	
PARALLELS_API_VER_5 (p. 1670)	
PARALLELS_API_VER_6 (p. 1670)	
PARALLELS_API_VER_7 (p. 1671)	
PARALLELS_SDK_LIB_MAJ (p. 1671)	Current major version of SDK library
PVM_GET_SECURITY_LEVEL (p. 1671)	
PVS_GET_GUEST_NUMBER (p. 1671)	
PVS_GET_GUEST_TYPE (p. 1671)	
PVS_MAKE_GUEST_VER (p. 1672)	
SET_FS_EXECUTE_DISABLED (p. 1672)	
SET_FS_EXECUTE_ENABLED (p. 1672)	
SET_FS_READ_DISABLED (p. 1672)	
SET_FS_READ_ENABLED (p. 1673)	
SET_FS_WRITE_DISABLED (p. 1673)	
SET_FS_WRITE_ENABLED (p. 1673)	
TO_PARALLELS_API_VER (p. 1673)	Converts major and minor versions of the API to the format understood by SDK library.

___PARALLELS_DISK_SDK_DEFINITIONS___ Macro

Syntax

```
#define ___PARALLELS_DISK_SDK_DEFINITIONS___
```

File

PrlDisk.h

__PARALLELS_API_COMMANDS_FLAGS_H__ Macro

Syntax

```
#define __PARALLELS_API_COMMANDS_FLAGS_H__
```

File

PrlCommandsFlags.h

__PARALLELS_API_CORE_H__ Macro

Syntax

```
#define __PARALLELS_API_CORE_H__
```

File

PrlApiCore.h

__PARALLELS_API_CT_H__ Macro

Syntax

```
#define __PARALLELS_API_CT_H__
```

File

PrlApiCt.h

__PARALLELS_API_DISPATCHER_H__ Macro

Syntax

```
#define __PARALLELS_API_DISPATCHER_H__
```

File

PrlApiDisp.h

__PARALLELS_API_ENUMS_H__ Macro

Syntax

```
#define __PARALLELS_API_ENUMS_H__
```

File

PrlEnums.h

__PARALLELS_API_ERRORS_H__ Macro

Syntax

```
#define __PARALLELS_API_ERRORS_H__
```

File

PrlErrors.h

__PARALLELS_API_ERRORS_VALUES_H__ Macro

Syntax

```
#define __PARALLELS_API_ERRORS_VALUES_H__
```

File

PrlErrorsValues.h

__PARALLELS_API_EVENTS_VALUES_H__ Macro

Syntax

```
#define __PARALLELS_API_EVENTS_VALUES_H__
```

File

PrlEventsValues.h

__PARALLELS_API_FS_H__ Macro

Syntax

```
#define __PARALLELS_API_FS_H__
```

File

PrlFs.h

__PARALLELS_API_HEADER_H__ Macro

Syntax

```
#define __PARALLELS_API_HEADER_H__
```

File

Parallels.h

__PARALLELS_API_KEYS_H__ Macro

Syntax

```
#define __PARALLELS_API_KEYS_H__
```

File

PrlKeys.h

__PARALLELS_API_MANAGEMENT_H__ Macro

Syntax

```
#define __PARALLELS_API_MANAGEMENT_H__
```

File

PrlApi.h

__PARALLELS_API_NET_H__ Macro

Syntax

```
#define __PARALLELS_API_NET_H__
```

File

PrlApiNet.h

__PARALLELS_API_OSES_H__ Macro

Syntax

```
#define __PARALLELS_API_OSES_H__
```

File

PrlOses.h

__PARALLELS_API_PORTS_H__ Macro

Syntax

```
#define __PARALLELS_API_PORTS_H__
```

File

PrlPorts.h

__PARALLELS_API_STATISTIC_H__ Macro

Syntax

```
#define __PARALLELS_API_STATISTIC_H__
```

File

PrlApiStat.h

__PARALLELS_API_TYPES_H__ Macro

Syntax

```
#define __PARALLELS_API_TYPES_H__
```

File

PrlTypes.h

__PARALLELS_API_VM_H__ Macro

Syntax

```
#define __PARALLELS_API_VM_H__
```

File

PrlApiVm.h

__PARALLELS_IO_STRUCTS_H__ Macro

Syntax

```
#define __PARALLELS_IO_STRUCTS_H__
```

File

PrlIOStructs.h

__PARALLELS_PERF_COUNTERS_H__ Macro

Syntax

```
#define __PARALLELS_PERF_COUNTERS_H__
```

File

PrlPerfCounters.h

__PRL_PLUGINS_CLASSES_DEFINED__ Macro

Syntax

```
#define __PRL_PLUGINS_CLASSES_DEFINED__
```

File

PrlPluginClasses.h

__PRLAPIVERSION_H__ Macro

Syntax

```
#define __PRLAPIVERSION_H__
```

File

PrlApiVersion.h

CAPDEFAULTMASK Macro

Syntax

```
#define CAPDEFAULTMASK ( \
    PCC_CHOWN         | \
    PCC_DAC_OVERRIDE    | \
    PCC_DAC_READ_SEARCH    | \
    PCC_FOWNER         | \
    PCC_FSETID         | \
    PCC_KILL         | \
    PCC_SETGID         | \
    PCC_SETUID         | \
    PCC_LINUX_IMMUTABLE    | \
    PCC_NET_BIND_SERVICE    | \
    PCC_NET_BROADCAST    | \
    PCC_NET_RAW         | \
    PCC_IPC_LOCK         | \
    PCC_IPC_OWNER         | \
    PCC_SYS_CHROOT         | \
    PCC_SYS_PTRACE         | \
    PCC_SYS_BOOT         | \
    PCC_SYS_NICE         | \
    PCC_SYS_RESOURCE    | \
    PCC_SYS_TTY_CONFIG    | \
    PCC_MKNOD         | \
    PCC_LEASE         | \
    PCC_VE_ADMIN         | \
    PCC_SETPCAP         | \
    PCC_SETFCAP         | \
    PCC_AUDIT_WRITE    )
```

File

PrlEnums.h

FROM_PARALLELS_API_VER Macro

Converts API version from the library native format to major and minor values.

Syntax

```
#define FROM_PARALLELS_API_VER(version, majorV, minorV) \
     majorV = ((PRL_UINT32)version) >> 16; \
     minorV = ((PRL_UINT32)version) & 0x0000FFFF
```

File

PrlApiVersion.h

FS_PERM_EXECUTE Macro

Syntax

```
#define FS_PERM_EXECUTE 0x1
```

File

PrlFs.h

FS_PERM_READ Macro

Syntax

```
#define FS_PERM_READ 0x2
```

File

PrlFs.h

FS_PERM_WRITE Macro

Syntax

```
#define FS_PERM_WRITE 0x4
```

File

PrlFs.h

GUID_CLS_BASE_STR Macro

Syntax

```
#define GUID_CLS_BASE_STR "{823067ea-8e15-474f-a3ae-c6a068461256}"
```

File

PrlPluginClasses.h

IS_ANDROID Macro

Syntax

```
#define IS_ANDROID(x) (!((x) >> 8) ^ PVS_GUEST_TYPE_ANDROID))
```

File

PrlOses.h

IS_CHROMEOS Macro

Syntax

```
#define IS_CHROMEOS(x) (!((x) >> 8) ^ PVS_GUEST_TYPE_CHROMEOS))
```

File

PrlOses.h

IS_FREEBSD Macro

Syntax

```
#define IS_FREEBSD(x) (!((x) >> 8) ^ PVS_GUEST_TYPE_FREEBSD))
```

File

PrlOses.h

IS_FS_EXECUTE_PERMITTED Macro

Syntax

```
#define IS_FS_EXECUTE_PERMITTED(x) ( (x) & FS_PERM_EXECUTE ?1:0 )
```

File

PrlFs.h

IS_FS_READ_PERMITTED Macro

Syntax

```
#define IS_FS_READ_PERMITTED(x) ( (x) & FS_PERM_READ ?1:0 )
```

File

PrlFs.h

IS_FS_WRITE_PERMITTED Macro

Syntax

```
#define IS_FS_WRITE_PERMITTED(x) ( (x) & FS_PERM_WRITE ?1:0 )
```

File

PrlFs.h

IS_LINUX Macro

Syntax

```
#define IS_LINUX(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_LINUX) ||    \
                                    IS_CHROMEOS(x) || IS_ANDROID(x))
```

File

PrlOses.h

IS_MACOS Macro

Syntax

```
#define IS_MACOS(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_MACOS))
```

File

PrlOses.h

IS_MSDOS Macro

Syntax

```
#define IS_MSDOS(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_MSDOS))
```

File

PrlOses.h

IS_NETWARE Macro

Syntax

```
#define IS_NETWARE(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_NETWARE))
```

File

PrlOses.h

IS_OS2 Macro

Syntax

```
#define IS_OS2(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_OS2))
```

File

PrlOses.h

IS_OTHER Macro

Syntax

```
#define IS_OTHER(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_OTHER))
```

File

PrlOses.h

IS_SOLARIS Macro

Syntax

```
#define IS_SOLARIS(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_SOLARIS))
```

File

PrlOses.h

IS_VALID_ANDROID_VERSION Macro

Syntax

```
#define IS_VALID_ANDROID_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_ANDROID_2_x &&                          \
      (ver) <= PVS_GUEST_VER_ANDROID_LAST) ||                        \
      (ver) == PVS_GUEST_VER_ANDROID_OTHER)
```

File

PrlOses.h

IS_VALID_CHROMEOS_VERSION Macro

Syntax

```
#define IS_VALID_CHROMEOS_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_CHROMEOS_1x &&                          \
      (ver) <= PVS_GUEST_VER_CHROMEOS_LAST) ||                       \
      (ver) == PVS_GUEST_VER_CHROMEOS_OTHER)
```

File

PrlOses.h

IS_VALID_FREEBSD_VERSION Macro

Syntax

```
#define IS_VALID_FREEBSD_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_BSD_4X && (ver) <= PVS_GUEST_VER_BSD_LAST) ||    \
     (ver) == PVS_GUEST_VER_BSD_OTHER)
```

File

PrlOses.h

IS_VALID_LINUX_VERSION Macro

Syntax

```
#define IS_VALID_LINUX_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_LIN_REDHAT && (ver) <= PVS_GUEST_VER_LIN_LAST) ||\
     (ver) == PVS_GUEST_VER_LIN_OTHER || IS_VALID_CHROMEOS_VERSION(ver) ||\
     IS_VALID_ANDROID_VERSION(ver))
```

File

PrlOses.h

IS_VALID_MACOS_VERSION Macro

Syntax

```
#define IS_VALID_MACOS_VERSION(ver) \
    ((ver) >= PVS_GUEST_VER_MACOS_TIGER && (ver) <= PVS_GUEST_VER_MACOS_LAST)
```

File

PrlOses.h

IS_VALID_MSDOS_VERSION Macro

Syntax

```
#define IS_VALID_MSDOS_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_DOS_MS622 && (ver) <= PVS_GUEST_VER_DOS_LAST)||\
     (ver) == PVS_GUEST_VER_DOS_OTHER)
```

File

PrlOses.h

IS_VALID_NETWARE_VERSION Macro

Syntax

```
#define IS_VALID_NETWARE_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_NET_4X && (ver) <= PVS_GUEST_VER_NET_LAST) ||    \
     (ver) == PVS_GUEST_VER_NET_OTHER)
```

File

PrlOses.h

IS_VALID_OS2_VERSION Macro

Syntax

```
#define IS_VALID_OS2_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_OS2_WARP3 && (ver) <= PVS_GUEST_VER_OS2_LAST)||\
     (ver) == PVS_GUEST_VER_OS2_OTHER)
```

File

PrlOses.h

IS_VALID_OTHER_OS_VERSION Macro

Syntax

```
#define IS_VALID_OTHER_OS_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_OTH_QNX && (ver) <= PVS_GUEST_VER_OTH_LAST) ||    \
     (ver) == PVS_GUEST_VER_OTH_OTHER)
```

File

PrlOses.h

IS_VALID_SOLARIS_VERSION Macro

Syntax

```
#define IS_VALID_SOLARIS_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_SOL_9 && (ver) <= PVS_GUEST_VER_SOL_LAST) ||    \
     (ver) == PVS_GUEST_VER_SOL_OTHER)
```

File

PrlOses.h

IS_VALID_WINDOWS_VERSION Macro

Syntax

```
#define IS_VALID_WINDOWS_VERSION(ver) \
    (((ver) >= PVS_GUEST_VER_WIN_311 && (ver) <= PVS_GUEST_VER_WIN_LAST) ||    \
     (ver) == PVS_GUEST_VER_WIN_OTHER)
```

File

PrlOses.h

IS_WIN_VER_BELOW Macro

PVS_GUEST_VER_WIN_OTHER (p. 1618) is used for legacy guest OS (9x and below), but has maximal number 0xFF.

Syntax

```
#define IS_WIN_VER_BELOW(ver1, ver2) ((ver1 < ver2) || (ver1 == PVS_GUEST_VER_WIN_OTHER))
```

File

PrlOses.h

IS_WINDOWS Macro

Syntax

```
#define IS_WINDOWS(x) (!(((x) >> 8) ^ PVS_GUEST_TYPE_WINDOWS))
```

File

PrlOses.h

MAX_PARALLEL_COUNT Macro

Syntax

```
#define MAX_PARALLEL_COUNT 3
```

File

PrlEnums.h

MAX_USB_PRINTER_COUNT Macro

Now we have 128 usb1.1 devices, but on 2 from it mapped mouse and keyboard

Syntax

```
#define MAX_USB_PRINTER_COUNT 126 // Now we have 128 usb1.1 devices, but on 2 from it mapped mouse and keyboard
```

File

PrlEnums.h

PARALLELS_API_VER Macro

Current Virtuozzo API version.

Syntax

```
#define PARALLELS_API_VER PARALLELS_API_VER_7
```

File

PrlApiVersion.h

PARALLELS_API_VER_1 Macro

Syntax

```
#define PARALLELS_API_VER_1 TO_PARALLELS_API_VER(0,1)
```

File

PrlApiVersion.h

PARALLELS_API_VER_2 Macro

Syntax

```
#define PARALLELS_API_VER_2 TO_PARALLELS_API_VER(2,0)
```

File

PrlApiVersion.h

PARALLELS_API_VER_3 Macro

Syntax

```
#define PARALLELS_API_VER_3 TO_PARALLELS_API_VER(3,0)
```

File

PrlApiVersion.h

PARALLELS_API_VER_4 Macro

Syntax

```
#define PARALLELS_API_VER_4 TO_PARALLELS_API_VER(4,0)
```

File

PrlApiVersion.h

PARALLELS_API_VER_5 Macro

Syntax

```
#define PARALLELS_API_VER_5 TO_PARALLELS_API_VER(5,0)
```

File

PrlApiVersion.h

PARALLELS_API_VER_6 Macro

Syntax

```
#define PARALLELS_API_VER_6 TO_PARALLELS_API_VER(6,0)
```

File

PrlApiVersion.h

PARALLELS_API_VER_7 Macro

Syntax

```
#define PARALLELS_API_VER_7 TO_PARALLELS_API_VER(7,0)
```

File

PrlApiVersion.h

PARALLELS_SDK_LIB_MAJ Macro

Current major version of SDK library

Syntax

```
#define PARALLELS_SDK_LIB_MAJ "7"
```

File

PrlApiVersion.h

PVM_GET_SECURITY_LEVEL Macro

Syntax

```
#define PVM_GET_SECURITY_LEVEL(flags) \
    (PRL_SECURITY_LEVEL)(flags&(PSL_LOW_SECURITY|PSL_NORMAL_SECURITY|PSL_HIGH_SECURITY))
```

File

PrlEnums.h

PVS_GET_GUEST_NUMBER Macro

Syntax

```
#define PVS_GET_GUEST_NUMBER(v) ((v) & 0xff)
```

File

PrlOses.h

PVS_GET_GUEST_TYPE Macro

Syntax

```
#define PVS_GET_GUEST_TYPE(v) (((v) >> 8) & 0xff)
```

File

PrlOses.h

PVS_MAKE_GUEST_VER Macro

Syntax

```
#define PVS_MAKE_GUEST_VER(t,n) ((((t) << 8) & 0xff00) | ((n) & 0xff))
```

File

PrlOses.h

SET_FS_EXECUTE_DISABLED Macro

Syntax

```
#define SET_FS_EXECUTE_DISABLED(x) (x) & ~FS_PERM_EXECUTE;
```

File

PrlFs.h

SET_FS_EXECUTE_ENABLED Macro

Syntax

```
#define SET_FS_EXECUTE_ENABLED(x) (x) | FS_PERM_EXECUTE;
```

File

PrlFs.h

SET_FS_READ_DISABLED Macro

Syntax

```
#define SET_FS_READ_DISABLED(x) (x) & ~FS_PERM_READ;
```

File

PrlFs.h

SET_FS_READ_ENABLED Macro

Syntax

```
#define SET_FS_READ_ENABLED(x) (x) | FS_PERM_READ;
```

File

PrlFs.h

SET_FS_WRITE_DISABLED Macro

Syntax

```
#define SET_FS_WRITE_DISABLED(x) (x) & ~FS_PERM_WRITE;
```

File

PrlFs.h

SET_FS_WRITE_ENABLED Macro

Syntax

```
#define SET_FS_WRITE_ENABLED(x) (x) | FS_PERM_WRITE;
```

File

PrlFs.h

TO_PARALLELS_API_VER Macro

Converts major and minor versions of the API to the format understood by SDK library.

Syntax

```
#define TO_PARALLELS_API_VER(majorV, minorV) \
     ((PRL_UINT32)(((PRL_UINT32)majorV) << 16 | ((PRL_UINT16)minorV)))
```

File

PrlApiVersion.h

Index

__

___PARALLELS_DISK_SDK_DEFINITIONS___ 1656

___PARALLELS_DISK_SDK_DEFINITIONS___ macro 1656

__PARALLELS_API_COMMANDS_FLAGS_H__ 1656

__PARALLELS_API_COMMANDS_FLAGS_H__ macro 1656

__PARALLELS_API_CORE_H__ 1657

__PARALLELS_API_CORE_H__ macro 1657

__PARALLELS_API_CT_H__ 1657

__PARALLELS_API_CT_H__ macro 1657

__PARALLELS_API_DISPATCHER_H__ 1657

__PARALLELS_API_DISPATCHER_H__ macro 1657

__PARALLELS_API_ENUMS_H__ 1657

__PARALLELS_API_ENUMS_H__ macro 1657

__PARALLELS_API_ERRORS_H__ 1657

__PARALLELS_API_ERRORS_H__ macro 1657

__PARALLELS_API_ERRORS_VALUES_H__ 1658

__PARALLELS_API_ERRORS_VALUES_H__ macro 1658

__PARALLELS_API_EVENTS_VALUES_H__ 1658

__PARALLELS_API_EVENTS_VALUES_H__ macro 1658

__PARALLELS_API_FS_H__ 1658

__PARALLELS_API_FS_H__ macro 1658

__PARALLELS_API_HEADER_H__ 1658

__PARALLELS_API_HEADER_H__ macro 1658

__PARALLELS_API_KEYS_H__ 1658

__PARALLELS_API_KEYS_H__ macro 1658

__PARALLELS_API_MANAGEMENT_H__ 1659

__PARALLELS_API_MANAGEMENT_H__ macro 1659

__PARALLELS_API_NET_H__ 1659

__PARALLELS_API_NET_H__ macro 1659

__PARALLELS_API_OSES_H__ 1659

__PARALLELS_API_OSES_H__ macro 1659

__PARALLELS_API_PORTS_H__ 1659

__PARALLELS_API_PORTS_H__ macro 1659

__PARALLELS_API_STATISTIC_H__ 1659

__PARALLELS_API_STATISTIC_H__ macro 1659

__PARALLELS_API_TYPES_H__ 1660

__PARALLELS_API_TYPES_H__ macro 1660

__PARALLELS_API_VM_H__ 1660

__PARALLELS_API_VM_H__ macro 1660

__PARALLELS_IO_STRUCTS_H__ 1660

__PARALLELS_IO_STRUCTS_H__ macro 1660

__PARALLELS_PERF_COUNTERS_H__ 1660

__PARALLELS_PERF_COUNTERS_H__ macro 1660

__PRL_DISK_PARAMETERS 963

__PRL_DISK_PARAMETERS structure 963

__PRL_IMAGE_PARAMETERS 964

__PRL_IMAGE_PARAMETERS structure 964

__PRL_PLUGINS_CLASSES_DEFINED__ 1660

__PRL_PLUGINS_CLASSES_DEFINED__ macro 1660

__PRL_SNAPSHOT_ELEMENT 964

__PRL_SNAPSHOT_ELEMENT structure 964

__PRLAPIVERSION_H__ 1661

__PRLAPIVERSION_H__ macro 1661

_IPluginInfo 964

_IPluginInfo structure 964

_PRL_ADAPTER_INDEX 965

_PRL_ADAPTER_INDEX enumeration 965

_PRL_ALLOWED_VM_COMMAND 965

_PRL_ALLOWED_VM_COMMAND enumeration 965

_PRL_API_COMMAND_FLAGS 966

_PRL_API_COMMAND_FLAGS enumeration 966

_PRL_API_INIT_FLAGS 967

_PRL_API_INIT_FLAGS enumeration 967

_PRL_API_METHOD 967

_PRL_API_METHOD structure 967

- about _PRL_API_METHOD structure 967
- MethodAPIVersion 969
- MethodIsAsync 969
- MethodIsDeprecated 969
- MethodIsSrv 969
- MethodName 969
- MethodPtr 970
- PRL_API_METHOD_PTR 970
- PRL_CONST_API_METHOD_PTR 970
- PRL_CONST_GUID_PTR 970

PRL_FILE_DESC 970
PRL_GUID 970
PRL_GUID_PTR 971
PRL_RESULT 970, 971
PRL_RESULT_PTR 971
_PRL_API_METHOD::_PRL_GUID 971
_PRL_API_METHOD::_PRL_GUID structure 971
_PRL_API_METHOD::MethodAPIVersion 969
_PRL_API_METHOD::MethodIsAsync 969
_PRL_API_METHOD::MethodIsDeprecated 969
_PRL_API_METHOD::MethodIsSrv 969
_PRL_API_METHOD::MethodName 969
_PRL_API_METHOD::MethodPtr 970
_PRL_API_METHOD::PRL_API_METHOD_PTR 970
_PRL_API_METHOD::PRL_CONST_API_METHOD_PTR 970
_PRL_API_METHOD::PRL_CONST_GUID_PTR 970
_PRL_API_METHOD::PRL_FILE_DESC 970
_PRL_API_METHOD::PRL_GUID 970
_PRL_API_METHOD::PRL_GUID_PTR 971
_PRL_API_METHOD::PRL_RESULT 970, 971
_PRL_API_METHOD::PRL_RESULT_PTR 971
_PRL_APP_IN_DOCK_MODE 971
_PRL_APP_IN_DOCK_MODE enumeration 971
_PRL_APPLIANCE_DOWNLOAD_STATUS 972
_PRL_APPLIANCE_DOWNLOAD_STATUS enumeration 972
_PRL_APPLICATION_MODE 972
_PRL_APPLICATION_MODE enumeration 972
_PRL_BEGIN_VM_BACKUP_FLAGS 972
_PRL_BEGIN_VM_BACKUP_FLAGS enumeration 972
_PRL_BIOS_TYPE 973
_PRL_BIOS_TYPE enumeration 973
_PRL_CHS 973
_PRL_CHS structure 973
_PRL_CLONE_VM_FLAGS 973
_PRL_CLONE_VM_FLAGS enumeration 973
_PRL_CLUSTERED_DEVICE_SUBTYPE 974
_PRL_CLUSTERED_DEVICE_SUBTYPE enumeration 974
_PRL_COMMIT_ENCRYPTION_FLAGS 974
_PRL_COMMIT_ENCRYPTION_FLAGS enumeration 974
_PRL_COMPACT_MODE 974
_PRL_COMPACT_MODE enumeration 974
_PRL_CONVERT_VIRTUAL_DISK 975
_PRL_CONVERT_VIRTUAL_DISK enumeration 975
_PRL_COPY_CT_TEMPLATE_FLAGS 975
_PRL_COPY_CT_TEMPLATE_FLAGS enumeration 975
_PRL_CORNER_ACTIONS 976
_PRL_CORNER_ACTIONS enumeration 976
_PRL_CPU_FEATURES_EX 976
_PRL_CPU_FEATURES_EX enumeration 976
_PRL_CPU_HVT 976
_PRL_CPU_HVT enumeration 976
_PRL_CPU_MODE 977
_PRL_CPU_MODE enumeration 977
_PRL_CPULIMIT_TYPE 977
_PRL_CPULIMIT_TYPE enumeration 977
_PRL_CPUS_COUNT 977
_PRL_CPUS_COUNT enumeration 977
_PRL_CPUUNITS 978
_PRL_CPUUNITS enumeration 978
_PRL_CREATE_DISK_IMAGE_FLAGS 978
_PRL_CREATE_DISK_IMAGE_FLAGS enumeration 978
_PRL_CREATE_SNAPSHOT_FLAGS 978
_PRL_CREATE_SNAPSHOT_FLAGS enumeration 978
_PRL_CT_CAPMASK 979
_PRL_CT_CAPMASK enumeration 979
_PRL_CT_FEATURES 980
_PRL_CT_FEATURES enumeration 980
_PRL_CT_REINSTALL_OPTIONS 980
_PRL_CT_REINSTALL_OPTIONS enumeration 980
_PRL_CT_RESOURCE 980
_PRL_CT_RESOURCE enumeration 980
_PRL_CT_TEMPLATE_TYPE 981
_PRL_CT_TEMPLATE_TYPE enumeration 981
_PRL_DATA_STATISTIC_SEGMENTS 981
_PRL_DATA_STATISTIC_SEGMENTS enumeration 981
_PRL_DELETE_SNAPSHOT_FLAGS 981
_PRL_DELETE_SNAPSHOT_FLAGS enumeration 981
_PRL_DEVICE_ACTION 982

_PRL_DEVICE_ACTION enumeration 982
_PRL_DEVICE_TYPE 982
_PRL_DEVICE_TYPE enumeration 982
_PRL_DISPLAY_BUFFER_FORMAT 983
_PRL_DISPLAY_BUFFER_FORMAT enumeration 983
_PRL_DISPLAY_CODEC_COMPRESSION 983
_PRL_DISPLAY_CODEC_COMPRESSION enumeration 983
_PRL_DISPLAY_CODEC_QUALITY 984
_PRL_DISPLAY_CODEC_QUALITY enumeration 984
_PRL_DISPLAY_CODEC_TYPE 984
_PRL_DISPLAY_CODEC_TYPE enumeration 984
_PRL_DISPLAY_CONFIG_MODE 984
_PRL_DISPLAY_CONFIG_MODE enumeration 984
_PRL_DISTRO_MEDIA_TYPE 985
_PRL_DISTRO_MEDIA_TYPE enumeration 985
_PRL_DOCK_ICON_TYPE 985
_PRL_DOCK_ICON_TYPE enumeration 985
_PRL_END_VM_BACKUP_FLAGS 986
_PRL_END_VM_BACKUP_FLAGS enumeration 986
_PRL_EVENT_ISSUER_TYPE 986
_PRL_EVENT_ISSUER_TYPE enumeration 986
_PRL_EVENT_TYPE 986
_PRL_EVENT_TYPE enumeration 986
_PRL_EXTERNAL_VM_TYPE 991
_PRL_EXTERNAL_VM_TYPE enumeration 991
_PRL_FEATURES_MATRIX 991
_PRL_FEATURES_MATRIX enumeration 991
_PRL_FILE_SYSTEM_ELEMENT_TYPE 992
_PRL_FILE_SYSTEM_ELEMENT_TYPE enumeration 992
_PRL_FILE_SYSTEM_FS_TYPE 992
_PRL_FILE_SYSTEM_FS_TYPE enumeration 992
_PRL_FILE_SYSTEM_TYPE 992
_PRL_FILE_SYSTEM_TYPE enumeration 992
_PRL_FIREWALL_DIRECTION 993
_PRL_FIREWALL_DIRECTION enumeration 993
_PRL_FIREWALL_POLICY 993
_PRL_FIREWALL_POLICY enumeration 993
_PRL_GENERIC_DEVICE_STATE 993
_PRL_GENERIC_DEVICE_STATE enumeration 993
_PRL_GENERIC_PCI_DEVICE_CLASS 994
_PRL_GENERIC_PCI_DEVICE_CLASS enumeration 994
_PRL_GET_SNAPSHOT_TREE_FLAGS 994
_PRL_GET_SNAPSHOT_TREE_FLAGS enumeration 994
_PRL_GET_VM_CONFIG_FLAGS 994
_PRL_GET_VM_CONFIG_FLAGS enumeration 994
_PRL_GET_VM_LIST_FLAGS 995
_PRL_GET_VM_LIST_FLAGS enumeration 995
_PRL_GRACEFUL_SHUTDOWN_CAPABILITIES 995
_PRL_GRACEFUL_SHUTDOWN_CAPABILITIES enumeration 995
_PRL_GRACEFUL_SHUTDOWN_KIND 995
_PRL_GRACEFUL_SHUTDOWN_KIND enumeration 995
_PRL_GUEST_DISTRO_VERSION_ENUM 996
_PRL_GUEST_DISTRO_VERSION_ENUM enumeration 996
_PRL_GUEST_OS_SUPPORT_TYPE 996
_PRL_GUEST_OS_SUPPORT_TYPE enumeration 996
_PRL_HANDLE_TYPE 996
_PRL_HANDLE_TYPE enumeration 996
_PRL_HARD_DISK_IFACE_MASK 998
_PRL_HARD_DISK_IFACE_MASK enumeration 998
_PRL_HARD_DISK_INTERNAL_FORMAT 999
_PRL_HARD_DISK_INTERNAL_FORMAT enumeration 999
_PRL_HOST_OS_TYPE 999
_PRL_HOST_OS_TYPE enumeration 999
_PRL_HW_INFO_NET_ADAPTER_TYPE 999
_PRL_HW_INFO_NET_ADAPTER_TYPE enumeration 999
_PRL_IMAGE_FORMAT 1000
_PRL_IMAGE_FORMAT enumeration 1000
_PRL_INSTALL_APPLIANCE_FLAGS 1000
_PRL_INSTALL_APPLIANCE_FLAGS enumeration 1000
_PRL_IO_CLIENT_FLAGS 837
_PRL_IO_CLIENT_FLAGS enumeration 837
_PRL_IO_DISCONNECT_REASON 838
_PRL_IO_DISCONNECT_REASON enumeration 838
_PRL_IO_DISPLAY_SCREEN_SIZE 838
_PRL_IO_DISPLAY_SCREEN_SIZE structure 838
_PRL_IO_DPI_DISPLAY_SCREEN_SIZE 839
_PRL_IO_DPI_DISPLAY_SCREEN_SIZE structure 839
about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE structure

839

display 844

dpi 844

PRL_IO_DISPLAY_GAMMA 844

PRL_IO_DISPLAY_GAMMA_PTR 845

PRL_IO_DISPLAY_PALETTE 845

PRL_IO_DISPLAY_PALETTE_PTR 845

PRL_IO_DISPLAY_SCREEN_REGION 845

PRL_IO_DISPLAY_SCREEN_REGION_PTR 845

PRL_IO_DPI_DISPLAY_SCREEN_SIZE_PTR 845

PRL_IO_SCREEN_BUFFER_REGION 846

PRL_IO_SCREEN_BUFFER_REGION_PTR 846

PRL_IO_SCREEN_REGION 846

PRL_IO_SCREEN_REGION_PTR 846

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_GAMMA
846

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_GAMMA
structure 846

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_PALETTE
847

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_PALETTE
structure 847

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION
847

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION
structure 847

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION structure 847

success 851

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX
851

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX
structure 851

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX structure 851

PRL_IO_DESKTOP_UTILITIES_STATE 856

PRL_IO_DESKTOP_UTILITIES_STATE_PTR 857

PRL_IO_GRACEFUL_SHUTDOWN 857

PRL_IO_GRACEFUL_SHUTDOWN_PTR 857

PRL_IO_KEYBOARD_LEDS 857

PRL_IO_KEYBOARD_LEDS_PTR 857

PRL_IO_KEYBOARD_SCANCODE 857

PRL_IO_KEYBOARD_SCANCODE_PTR 857

PRL_IO_LANGUAGE_HOTKEYS 858

PRL_IO_LANGUAGE_HOTKEYS_PTR 858

PRL_IO_MOUSE_EVENT 858

PRL_IO_MOUSE_EVENT_EX 858

PRL_IO_MOUSE_EVENT_EX_PTR 858

PRL_IO_MOUSE_EVENT_EX2 858

PRL_IO_MOUSE_EVENT_EX2_PTR 859

PRL_IO_MOUSE_EVENT_PTR 859

PRL_IO_SCREEN_SURFACE 859

PRL_IO_SCREEN_SURFACE_PTR 859

success 856

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_DESKTOP_UTILITIES_STATE
859

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_DESKTOP_UTILITIES_STATE
structure 859

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_GRACEFUL_SHUTDOWN
860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_GRACEFUL_SHUTDOWN
structure 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_KEYBOARD_LEDS
860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_

RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_KEYBOARD_LEDS structure 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_KEYBOARD_SCANCODE 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_KEYBOARD_SCANCODE structure 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_LANGUAGE_HOTKEYS 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_LANGUAGE_HOTKEYS structure 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR 860

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR structure 860

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR structure 860

height 865

hotSpotX 865

hotSpotY 865

PRL_IO_AVAILABLE_DISPLAYS 866

PRL_IO_AVAILABLE_DISPLAYS_PTR 866

PRL_IO_DEVICE_USED 866

PRL_IO_DEVICE_USED_PTR 866

PRL_IO_GENERIC_BUFFER 866

PRL_IO_GENERIC_BUFFER_PTR 866

PRL_IO_MOUSE_CURSOR_PTR 866

PRL_IO_SLIDING_MOUSE 867

PRL_IO_SLIDING_MOUSE_PTR 867

width 865

x 865

y 865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_AVAILABLE_DISPLAYS 867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_AVAILABLE_DISPLAYS structure 867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_DEVICE_USED 867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_DEVICE_USED structure 867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_GENERIC_BUFFER 867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_GENERIC_BUFFER structure 867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_SLIDING_MOUSE 868

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_IO_SLIDING_MOUSE structure 868

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION 868

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION structure 868

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPL

AY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION
structure 868

PRL_TIS_VERSION_PTR 871

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION
871

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION
structure 871

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION
structure 871

PRL_TIS_INT_VERSION_PTR 874

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO
874

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO
structure 874

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO
structure 874

PRL_TIS_RECORD_INFO_PTR 877

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND
877

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND
structure 877

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND
structure 877

PRL_IO_TOOLS_UTILITY_COMMAND_PTR 880

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA
880

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA
structure 880

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA
structure 880

PRL_IO_DYNRES_TOOLS_STATUS 884

PRL_IO_DYNRES_TOOLS_STATUS_PTR 884

PRL_IO_TOOLS_SIA_DATA_PTR 884

PRL_PROXY_CLIENT_INFO 885

PRL_PROXY_CLIENT_INFO_PTR 885

PRL_PROXY_DATA_KEY 885

PRL_PROXY_DATA_KEY_PTR 886

PRL_PROXY_GEO_INFO 886

PRL_PROXY_GEO_INFO_PTR 886

PRL_PROXY_HOST_INFO 886

PRL_PROXY_HOST_INFO_EXT 887

PRL_PROXY_HOST_INFO_EXT_PTR 887

PRL_PROXY_HOST_INFO_PTR 887

PRL_PROXY_REGISTRATION_INFO 887

PRL_PROXY_REGISTRATION_INFO_PTR 888

PRL_PROXY_USER_DATA 888

PRL_PROXY_USER_DATA_PTR 889

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_DYNRES_TOOLS_STATUS
889

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_DYNRES_TOOLS_STATUS
structure 889

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO
889

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO
structure 889

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO
structure 889

filenameSize 892

PRL_IO_ATTACH_REQUEST 893

PRL_IO_ATTACH_REQUEST_PTR 893

PRL_IO_ATTACH_RESPONSE 893

PRL_IO_ATTACH_RESPONSE_PTR 894

PRL_IO_AUTH_EXEC_REQUEST 894

PRL_IO_AUTH_EXEC_REQUEST_PTR 894

PRL_IO_AUTH_REQUEST 894

PRL_IO_AUTH_REQUEST_PTR 894

PRL_IO_AUTH_RESPONSE 895

PRL_IO_AUTH_RESPONSE_PTR 895

PRL_IO_MEMORY_INFO_PTR 895

ramOffset 892

ramSize 892

vramOffset 893

vramSize 893

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_ATTACH_REQUEST
895

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_ATTACH_REQUEST
structure 895

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_ATTACH_RESPONSE
896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_ATTACH_RESPONSE
structure 896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_EXEC_REQUEST
896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_EXEC_REQUEST
structure 896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_I

O_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_REQUEST
896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_REQUEST
structure 896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_RESPONSE
896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_AUTH_RESPONSE
structure 896

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE
897

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE
structure 897

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE
structure 897

display 900

height 900

offset 900

PRL_CONST_CPULIMIT_DATA_PTR 901

PRL_CONST_CVSRC_OPEN_MODE 901

PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR 901

PRL_CONST_IOLIMIT_DATA_PTR 901

PRL_CPULIMIT_DATA 901

PRL_CPULIMIT_DATA_PTR 902

PRL_CVSRC_OPEN_MODE 902

PRL_CVSRC_OPEN_MODE_PTR 902

PRL_GET_VM_CONFIG_PARAM_DATA 902

PRL_GET_VM_CONFIG_PARAM_DATA_PTR 902

PRL_IO_SCREEN_UPDATE_PTR 903

PRL_IO_SSO_CLIENT_VM_PACKET 903

PRL_IO_SSO_CLIENT_VM_PACKET_PTR 903

PRL_IOLIMIT_DATA 903

PRL_IOLIMIT_DATA_PTR 903

PRL_STAT_NET_TRAFFIC 904

PRL_STAT_NET_TRAFFIC_PTR 904

width 900

x 900

y 900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_CPULIMIT_DATA
904

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_CPULIMIT_DATA
structure 904

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_CVSRC_OPEN_MODE
904

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_

PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_CVSRC_OPEN_MODE
structure 904

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_GET_VM_CONFIG_PARAM_DATA
905

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_GET_VM_CONFIG_PARAM_DATA
structure 905

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT
905

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT
structure 905

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT
structure 905

PRL_IO_CHANNEL_STAT_PTR 906

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK
906

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK
structure 906

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK
structure 906

PRL_CONST_CPU_FEATURES_MASK_PTR 907

PRL_CPU_FEATURES_MASK_PTR 907

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES
907

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES
structure 907

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES
structure 907

PRL_CPU_FEATURES_PTR 908

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_I

O_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO
908

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO
structure 908

about _PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO
structure 908

PRL_CONST_MEMGUARANTEE_DATA_PTR 909

PRL_HOST_OS_INFO_PTR 909

PRL_MEMGUARANTEE_DATA 910

PRL_MEMGUARANTEE_DATA_PTR 910

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::_PRL_MEMGUARANTEE_DATA
910

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::_PRL_MEMGUARANTEE_DATA
structure 910

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_CONST_MEMGUARANTEE_DATA_PTR
909

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_HOST_OS_INFO_PTR
909

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::_PRL_HOST_OS_INFO::PRL_MEMGUARANTEE_DATA
910

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::_PRL_CPU_FEATURES::PRL_CPU_FEATURES_PTR
908

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::PRL_CONST_CPU_FEATURES_MASK_PTR
907

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_

PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::_PRL_CPU_FEATURES_MASK::PRL_CPU_FEATURES_MASK_PTR
907

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_CHANNEL_STAT::PRL_IO_CHANNEL_STAT_PTR
906

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_SSO_CLIENT_VM_PACKET
910

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IO_SSO_CLIENT_VM_PACKET
structure 910

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IOLIMIT_DATA
911

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_IOLIMIT_DATA
structure 911

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_STAT_NET_TRAFFIC
911

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::_PRL_STAT_NET_TRAFFIC
structure 911

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::display
900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::height
900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::offset
900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_CPULIMIT_DATA_PTR
901

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_CVSRC_OPEN_MODE
901

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_GET_VM_CONFIG_PARAM_DATA_PTR
901

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOL

UTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CONST_IOLIMIT_DATA_PTR
901

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CPULIMIT_DATA
901

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CPULIMIT_DATA_PTR
902

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CVSRC_OPEN_MODE
902

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_CVSRC_OPEN_MODE_PTR
902

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_GET_VM_CONFIG_PARAM_DATA
902

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_GET_VM_CONFIG_PARAM_DATA_PTR
902

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOL

UTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IO_SCREEN_UPDATE_PTR
903

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IO_SSO_CLIENT_VM_PACKET
903

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IO_SSO_CLIENT_VM_PACKET_PTR
903

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IOLIMIT_DATA
903

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_IOLIMIT_DATA_PTR
903

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_STAT_NET_TRAFFIC
904

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::PRL_STAT_NET_TRAFFIC_PTR
904

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOL

UTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::width
900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::x
900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::_PRL_IO_SCREEN_UPDATE::y
900

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::filenameSize
892

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_REQUEST
893

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_REQUEST_PTR
893

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_RESPONSE
893

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_ATTACH_RESPONSE_PTR
894

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_EXEC_REQUEST
894

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_EXEC_REQUEST_PTR
894

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_REQUEST
894

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_REQUEST_PTR
894

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_RESPONSE
895

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_AUTH_RESPONSE_PTR
895

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RE

CORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::PRL_IO_MEMORY_INFO_PTR
895

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::ramOffset
892

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::ramSize
892

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::vramOffset
893

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_IO_MEMORY_INFO::vramSize
893

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_CLIENT_INFO
911

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_CLIENT_INFO
structure 911

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_DATA_KEY
912

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_DATA_KEY
structure 912

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_GEO_INFO
912

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_GEO_INFO
structure 912

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_HOST_INFO
912

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_HOST_INFO
structure 912

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_HOST_INFO_EXT
913

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_HOST_INFO_EXT
structure 913

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_REGISTRATION_INFO
913

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_

PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_REGISTRATION_INFO structure 913

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_USER_DATA
914

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::_PRL_PROXY_USER_DATA structure 914

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_IO_DYNRES_TOOLS_STATUS
884

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_IO_DYNRES_TOOLS_STATUS_PTR
884

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_IO_TOOLS_SIA_DATA_PTR
884

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_CLIENT_INFO
885

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_CLIENT_INFO_PTR
885

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_DATA_KEY
885

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_DATA_KEY_PTR
886

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_GEO_INFO
886

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_GEO_INFO_PTR
886

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO
886

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO_EXT
887

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO_EXT_PTR
887

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_HOST_INFO_PTR
887

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOL

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOL

UTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_REGISTRATION_INFO
887

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_REGISTRATION_INFO_PTR
888

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_USER_DATA
888

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::_PRL_IO_TOOLS_SIA_DATA::PRL_PROXY_USER_DATA_PTR
889

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::_PRL_IO_TOOLS_UTILITY_COMMAND::PRL_IO_TOOLS_UTILITY_COMMAND_PTR
880

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::_PRL_TIS_RECORD_INFO::PRL_TIS_RECORD_INFO_PTR
877

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::_PRL_TIS_INT_VERSION::PRL_TIS_INT_VERSION_PTR
874

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::_PRL_TIS_VERSION::PRL_TIS_VERSION_PTR
871

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::height
865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::hotSpotX
865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::hotSpotY
865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_AVAILABLE_DISPLAYS
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_AVAILABLE_DISPLAYS_PTR
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_DEVICE_USED
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_DEVICE_USED_PTR
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_GENERIC_BUFFER
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_GENERIC_BUFFER_PTR
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_MOUSE_CURSOR_PTR
866

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_SLIDING_MOUSE
867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::PRL_IO_SLIDING_MOUSE_PTR
867

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::width 865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::x 865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_CURSOR::y 865

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT 915

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT structure 915

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT_EX 915

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT_EX structure 915

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT_EX2 915

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_MOUSE_EVENT_EX2 structure 915

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_SCREEN_SURFACE 916

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::_PRL_IO_SCREEN_SURFACE structure 916

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_DESKTOP_UTILITIES_STATE 856

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_DESKTOP_UTILITIES_STATE_PTR 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_GRACEFUL_SHUTDOWN 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_GRACEFUL_SHUTDOWN_PTR 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_LEDS 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_LEDS_PTR 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_SCANCODE 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_KEYBOARD_SCANCODE_PTR 857

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_LANGUAGE_HOTKEYS 858

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_LANGUAGE_HOTKEYS_PTR 858

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT 858

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX 858

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX_PTR

858

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX2 858

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_EX2_PTR 859

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_MOUSE_EVENT_PTR 859

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_SCREEN_SURFACE 859

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::PRL_IO_SCREEN_SURFACE_PTR 859

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION_EX::success 856

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_RESOLUTION_CONFIRMATION::success 851

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_SCREEN_REGION 916

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_DISPLAY_SCREEN_REGION structure 916

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_SCREEN_BUFFER_REGION 916

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_SCREEN_BUFFER_REGION structure 916

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_SCREEN_REGION 917

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::_PRL_IO_SCREEN_REGION structure 917

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::display 844

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::dpi 844

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_GAMMA 844

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_GAMMA_PTR 845

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_PALETTE 845

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_PALETTE_PTR 845

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_SCREEN_REGION 845

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DISPLAY_SCREEN_REGION_PTR 845

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_DPI_DISPLAY_SCREEN_SIZE_PTR 845

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_BUFFER_REGION 846

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_BUFFER_REGION_PTR 846

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_REGION 846

_PRL_IO_DPI_DISPLAY_SCREEN_SIZE::PRL_IO_SCREEN_REGION_PTR 846

_PRL_IO_SCREEN_SIZE 917

_PRL_IO_SCREEN_SIZE structure 917

_PRL_IO_STATE 917

_PRL_IO_STATE enumeration 917

_PRL_IOLIMIT_TYPE 1000

_PRL_IOLIMIT_TYPE enumeration 1000

_PRL_IOPRIO 1001

_PRL_IOPRIO enumeration 1001

_PRL_ISCSI_STORAGE_CMD_FLAGS 1001

_PRL_ISCSI_STORAGE_CMD_FLAGS enumeration 1001

_PRL_ISCSI_STORAGE_FSTYPE_FLAGS 1001

_PRL_ISCSI_STORAGE_FSTYPE_FLAGS enumeration 1001

_PRL_JOB_OPERATION_CODE 1002

_PRL_JOB_OPERATION_CODE enumeration 1002

_PRL_JOB_STATUS 1005

_PRL_JOB_STATUS enumeration 1005
_PRL_KEY 1005
_PRL_KEY enumeration 1005
_PRL_KEY_EVENT 1008
_PRL_KEY_EVENT enumeration 1008
_PRL_LICENSE_RESTRICTION_KEY 1009
_PRL_LICENSE_RESTRICTION_KEY enumeration 1009
_PRL_LOGIN_LOCAL_FLAGS 1010
_PRL_LOGIN_LOCAL_FLAGS enumeration 1010
_PRL_MASS_STORAGE_INTERFACE_TYPE 1011
_PRL_MASS_STORAGE_INTERFACE_TYPE enumeration 1011
_PRL_MEMGUARANTEE_TYPE 1011
_PRL_MEMGUARANTEE_TYPE enumeration 1011
_PRL_MESSAGE_TYPE_ENUM 1011
_PRL_MESSAGE_TYPE_ENUM enumeration 1011
_PRL_MOBILE_ADVANCED_AUTH_MODE 1012
_PRL_MOBILE_ADVANCED_AUTH_MODE enumeration 1012
_PRL_MOUNT_VIRTUAL_DISK 1012
_PRL_MOUNT_VIRTUAL_DISK enumeration 1012
_PRL_MOUSE_BUTTON 1012
_PRL_MOUSE_BUTTON enumeration 1012
_PRL_NET_ADAPTER_EMULATED_TYPE 1013
_PRL_NET_ADAPTER_EMULATED_TYPE enumeration 1013
_PRL_NET_ADAPTER_TYPE 1013
_PRL_NET_ADAPTER_TYPE enumeration 1013
_PRL_NET_VIRTUAL_NETWORK_TYPE 1013
_PRL_NET_VIRTUAL_NETWORK_TYPE enumeration 1013
_PRL_NETFILTER_MODE 1014
_PRL_NETFILTER_MODE enumeration 1014
_PRL_NETWORK_RATE_MPU 1014
_PRL_NETWORK_RATE_MPU enumeration 1014
_PRL_ONLINE_COMPACT_MODE 1014
_PRL_ONLINE_COMPACT_MODE enumeration 1014
_PRL_OPTIMIZE_MODIFIERS_MODE 1015
_PRL_OPTIMIZE_MODIFIERS_MODE enumeration 1015
_PRL_PARAM_FIELD_DATA_TYPE 1015
_PRL_PARAM_FIELD_DATA_TYPE enumeration 1015
_PRL_PORT_FORWARDING_TYPE 1015
_PRL_PORT_FORWARDING_TYPE enumeration 1015
_PRL_PRINTER_INTERFACE_TYPE 1016
_PRL_PRINTER_INTERFACE_TYPE enumeration 1016
_PRL_PRIVATE_EVENT_TYPE 1016
_PRL_PRIVATE_EVENT_TYPE enumeration 1016
_PRL_PROBLEM_REPORT_FLAGS 1018
_PRL_PROBLEM_REPORT_FLAGS enumeration 1018
_PRL_PROBLEM_REPORT_REASON 1019
_PRL_PROBLEM_REPORT_REASON enumeration 1019
_PRL_PROBLEM_REPORT_SCHEME 1019
_PRL_PROBLEM_REPORT_SCHEME enumeration 1019
_PRL_PROBLEM_REPORT_TYPE 1020
_PRL_PROBLEM_REPORT_TYPE enumeration 1020
_PRL_PROCESS_STATE_TYPE 1020
_PRL_PROCESS_STATE_TYPE enumeration 1020
_PRL_PRODUCT_ENUM 1021
_PRL_PRODUCT_ENUM enumeration 1021
_PRL_PROXY_CONNECTION_STATUS 1021
_PRL_PROXY_CONNECTION_STATUS enumeration 1021
_PRL_QUERY_ELEMENT_FLAGS 1021
_PRL_QUERY_ELEMENT_FLAGS enumeration 1021
_PRL_REGISTER_3PARTY_VM_FLAGS 1022
_PRL_REGISTER_3PARTY_VM_FLAGS enumeration 1022
_PRL_REGISTER_NEW_VM 1022
_PRL_REGISTER_NEW_VM enumeration 1022
_PRL_REGISTER_VM_FLAGS 1022
_PRL_REGISTER_VM_FLAGS enumeration 1022
_PRL_RESIZE_IMAGE_FLAGS 1023
_PRL_RESIZE_IMAGE_FLAGS enumeration 1023
_PRL_RUNNING_TASK_TYPE_ENUM 1023
_PRL_RUNNING_TASK_TYPE_ENUM enumeration 1023
_PRL_SCALE_VIEW_MODE 1023
_PRL_SCALE_VIEW_MODE enumeration 1023
_PRL_SECURITY_LEVEL 1024
_PRL_SECURITY_LEVEL enumeration 1024
_PRL_SERIAL_PORT_SOCKET_OPERATION_MODE 1024
_PRL_SERIAL_PORT_SOCKET_OPERATION_MODE enumeration 1024
_PRL_SERVICE_STATUS_ENUM 1024
_PRL_SERVICE_STATUS_ENUM enumeration 1024
_PRL_SET_USERPASSWD_FLAGS 1025

_PRL_SET_USERPASSWD_FLAGS enumeration 1025
_PRL_SOUND_CHANNEL_TYPE 1025
_PRL_SOUND_CHANNEL_TYPE enumeration 1025
_PRL_SRV_SHUTDOWN_FLAGS 1025
_PRL_SRV_SHUTDOWN_FLAGS enumeration 1025
_PRL_SWITCH_SNAPSHOT_FLAGS 1026
_PRL_SWITCH_SNAPSHOT_FLAGS enumeration 1026
_PRL_TASKBAR_STATES 1026
_PRL_TASKBAR_STATES enumeration 1026
_PRL_TIS_RECORD_FIELD 1027
_PRL_TIS_RECORD_FIELD enumeration 1027
_PRL_TIS_RECORD_STATE 1027
_PRL_TIS_RECORD_STATE enumeration 1027
_PRL_TOOLS_SIA_OPEN_IN_TYPES 1027
_PRL_TOOLS_SIA_OPEN_IN_TYPES enumeration 1027
_PRL_TOOLS_UTILITY_COMMAND_FLAG 1028
_PRL_TOOLS_UTILITY_COMMAND_FLAG enumeration 1028
_PRL_TOOLS_UTILITY_COMMAND_ID 1028
_PRL_TOOLS_UTILITY_COMMAND_ID enumeration 1028
_PRL_UNDO_DISKS_MODE 1029
_PRL_UNDO_DISKS_MODE enumeration 1029
_PRL_UPDATE_LICENSE_FLAGS 1029
_PRL_UPDATE_LICENSE_FLAGS enumeration 1029
_PRL_USB_DEVICE_AUTO_CONNECT_OPTION 1029
_PRL_USB_DEVICE_AUTO_CONNECT_OPTION enumeration 1029
_PRL_USB_DEVICE_CONNECT_REASON 1030
_PRL_USB_DEVICE_CONNECT_REASON enumeration 1030
_PRL_USB_DEVICE_TYPE 1030
_PRL_USB_DEVICE_TYPE enumeration 1030
_PRL_VCMMD_COMMAND_FLAGS 1032
_PRL_VCMMD_COMMAND_FLAGS enumeration 1032
_PRL_VERSION_ENUM 1032
_PRL_VERSION_ENUM enumeration 1032
_PRL_VIDEO_3D_ACCELERATION 1032
_PRL_VIDEO_3D_ACCELERATION enumeration 1032
_PRL_VIRTUAL_MACHINE_PROFILE 1033
_PRL_VIRTUAL_MACHINE_PROFILE enumeration 1033
_PRL_VIRTUAL_NETWORK_FLAGS 1033
_PRL_VIRTUAL_NETWORK_FLAGS enumeration 1033
_PRL_VM_ACCELERATION_LEVEL 923
_PRL_VM_ACCELERATION_LEVEL enumeration 923
_PRL_VM_ACCESS_FOR_OTHERS 923
_PRL_VM_ACCESS_FOR_OTHERS enumeration 923
_PRL_VM_ACTION 924
_PRL_VM_ACTION enumeration 924
_PRL_VM_ACTION_ON_STOP 924
_PRL_VM_ACTION_ON_STOP enumeration 924
_PRL_VM_ACTION_ON_WINDOW_CLOSE 924
_PRL_VM_ACTION_ON_WINDOW_CLOSE enumeration 924
_PRL_VM_AUTOSTART_OPTION 925
_PRL_VM_AUTOSTART_OPTION enumeration 925
_PRL_VM_AUTOSTOP_OPTION 925
_PRL_VM_AUTOSTOP_OPTION enumeration 925
_PRL_VM_BACKUP_FLAGS 925
_PRL_VM_BACKUP_FLAGS enumeration 925
_PRL_VM_BACKUP_TYPE 926
_PRL_VM_BACKUP_TYPE enumeration 926
_PRL_VM_BOOT_ORDER 927
_PRL_VM_BOOT_ORDER enumeration 927
_PRL_VM_COLOR 927
_PRL_VM_COLOR enumeration 927
_PRL_VM_COMMIT_FLAGS 927
_PRL_VM_COMMIT_FLAGS enumeration 927
_PRL_VM_CONFIG_SECTIONS 928
_PRL_VM_CONFIG_SECTIONS enumeration 928
_PRL_VM_CPULIMIT_TYPE 929
_PRL_VM_CPULIMIT_TYPE enumeration 929
_PRL_VM_DEV_EMULATION_TYPE 929
_PRL_VM_DEV_EMULATION_TYPE enumeration 929
_PRL_VM_EXEC_APP_FILE_DESCRIPTOR 929
_PRL_VM_EXEC_APP_FILE_DESCRIPTOR enumeration 929
_PRL_VM_EXEC_RUN_PROGRAM_MODES 930
_PRL_VM_EXEC_RUN_PROGRAM_MODES enumeration 930
_PRL_VM_LOCATION 930
_PRL_VM_LOCATION enumeration 930
_PRL_VM_MIGRATION_FLAGS 931
_PRL_VM_MIGRATION_FLAGS enumeration 931
_PRL_VM_MOUSE_CURSOR_STATE 932

_PRL_VM_MOUSE_CURSOR_STATE enumeration 932
_PRL_VM_MOUSE_STATE 932
_PRL_VM_MOUSE_STATE enumeration 932
_PRL_VM_NET_ADAPTER_TYPE 932
_PRL_VM_NET_ADAPTER_TYPE enumeration 932
_PRL_VM_NET_ADAPTER_WIFI_RULE 933
_PRL_VM_NET_ADAPTER_WIFI_RULE enumeration 933
_PRL_VM_NORMAL_START_FLAGS 933
_PRL_VM_NORMAL_START_FLAGS enumeration 933
_PRL_VM_ON_CRASH_ACTION 1033
_PRL_VM_ON_CRASH_ACTION enumeration 1033
_PRL_VM_ON_CRASH_OPTION 1033
_PRL_VM_ON_CRASH_OPTION enumeration 1033
_PRL_VM_PRIORITY 933
_PRL_VM_PRIORITY enumeration 933
_PRL_VM_REMOTE_DISPLAY_MODE 934
_PRL_VM_REMOTE_DISPLAY_MODE enumeration 934
_PRL_VM_SMARTGUARD_SCHEMA 934
_PRL_VM_SMARTGUARD_SCHEMA enumeration 934
_PRL_VM_START_LOGIN_MODE 934
_PRL_VM_START_LOGIN_MODE enumeration 934
_PRL_VM_START_MODE 935
_PRL_VM_START_MODE enumeration 935
_PRL_VM_STATISTICS_FLAGS 935
_PRL_VM_STATISTICS_FLAGS enumeration 935
_PRL_VM_STOP_FLAGS 935
_PRL_VM_STOP_FLAGS enumeration 935
_PRL_VM_STOP_MODE 936
_PRL_VM_STOP_MODE enumeration 936
_PRL_VM_STORE_VALUE_FLAGS 936
_PRL_VM_STORE_VALUE_FLAGS enumeration 936
_PRL_VM_TOOLS_STATE 936
_PRL_VM_TOOLS_STATE enumeration 936
_PRL_VM_TYPE 937
_PRL_VM_TYPE enumeration 937
_PRL_VM_TYPE_FLAGS 937
_PRL_VM_TYPE_FLAGS enumeration 937
_PRL_VM_WINDOW_MODE 937
_PRL_VM_WINDOW_MODE enumeration 937
_PRL_VMCTG_EVENT 1034
_PRL_VMCTG_EVENT enumeration 1034
_PRL_VTD_STATUS_CODE 1034
_PRL_VTD_STATUS_CODE enumeration 1034
_PRL_WINDOW_CORNERS 1035
_PRL_WINDOW_CORNERS enumeration 1035
_PRL_WINDOW_EDGE_MASK 1035
_PRL_WINDOW_EDGE_MASK enumeration 1035
_PrlPlugin 1035
_PrlPlugin structure 1035
_VIRTUAL_MACHINE_ADDITION_STATE 1103
_VIRTUAL_MACHINE_ADDITION_STATE enumeration 1103
_VIRTUAL_MACHINE_STATE 1103
_VIRTUAL_MACHINE_STATE enumeration 1103

A

About This Guide 1
API Debug Functions 744
Audio Playback 778

C

Callback Prototypes 779
CAPDEFAULTMASK 1661
CAPDEFAULTMASK macro 1661
Common Functions 727

D

DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_SET_ENCODING
enumeration member 1016
DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_STREAM_START
enumeration member 1016
DEPRECATED_PET_IO_CLI_AUDIO_OUTPUT_STREAM_STOP
enumeration member 1016
DMT_DISTRO enumeration member 985
DMT_LIVE_CD enumeration member 985
DMT_MBR enumeration member 985

E

event_type_to_string 744

event_type_to_string function 744

F

FROM_PARALLELS_API_VER 1661

FROM_PARALLELS_API_VER macro 1661

FS_PERM_EXECUTE 1662

FS_PERM_EXECUTE macro 1662

FS_PERM_READ 1662

FS_PERM_READ macro 1662

FS_PERM_WRITE 1662

FS_PERM_WRITE macro 1662

Functions by Handle 4

G

GUID_CLS_BASE_STR 1662

GUID_CLS_BASE_STR macro 1662

H

Handle Functions 747

handle_type_to_string 745

handle_type_to_string function 745

How to Use This Guide 2

I

Introduction 1

IOS_AUTH_FAILED enumeration member 917

IOS_CONNECTION_TIMEOUT enumeration member 917

IOS_DISABLED enumeration member 917

IOS_STARTED enumeration member 917

IOS_STOPPED enumeration member 917

IOS_UNKNOWN_VM_UUID enumeration member 917

IPluginInfo 1104

IPluginInfo structure 1104

IPluginInfoPtr 835

IPluginInfoPtr type 835

IS_ANDROID 1663

IS_ANDROID macro 1663

IS_CHROMEOS 1663

IS_CHROMEOS macro 1663

IS_FREEBSD 1663

IS_FREEBSD macro 1663

IS_FS_EXECUTE_PERMITTED 1663

IS_FS_EXECUTE_PERMITTED macro 1663

IS_FS_READ_PERMITTED 1663

IS_FS_READ_PERMITTED macro 1663

IS_FS_WRITE_PERMITTED 1664

IS_FS_WRITE_PERMITTED macro 1664

IS_LINUX 1664

IS_LINUX macro 1664

IS_MACOS 1664

IS_MACOS macro 1664

IS_MSDOS 1664

IS_MSDOS macro 1664

IS_NETWARE 1664

IS_NETWARE macro 1664

IS_OS2 1665

IS_OS2 macro 1665

IS_OTHER 1665

IS_OTHER macro 1665

IS_SOLARIS 1665

IS_SOLARIS macro 1665

IS_VALID_ANDROID_VERSION 1665

IS_VALID_ANDROID_VERSION macro 1665

IS_VALID_CHROMEOS_VERSION 1666

IS_VALID_CHROMEOS_VERSION macro 1666

IS_VALID_FREEBSD_VERSION 1666

IS_VALID_FREEBSD_VERSION macro 1666

IS_VALID_LINUX_VERSION 1666

IS_VALID_LINUX_VERSION macro 1666

IS_VALID_MACOS_VERSION 1666

IS_VALID_MACOS_VERSION macro 1666

IS_VALID_MSDOS_VERSION 1667

IS_VALID_MSDOS_VERSION macro 1667

IS_VALID_NETWARE_VERSION 1667

IS_VALID_NETWARE_VERSION macro 1667
IS_VALID_OS2_VERSION 1667
IS_VALID_OS2_VERSION macro 1667
IS_VALID_OTHER_OS_VERSION 1667
IS_VALID_OTHER_OS_VERSION macro 1667
IS_VALID_SOLARIS_VERSION 1668
IS_VALID_SOLARIS_VERSION macro 1668
IS_VALID_WINDOWS_VERSION 1668
IS_VALID_WINDOWS_VERSION macro 1668
IS_WIN_VER_BELOW 1668
IS_WIN_VER_BELOW macro 1668
IS_WINDOWS 1668
IS_WINDOWS macro 1668

K

Keyboard Control 778

M

Macros 1106
MAX_PARALLEL_COUNT 1669
MAX_PARALLEL_COUNT macro 1669
MAX_USB_PRINTER_COUNT 1669
MAX_USB_PRINTER_COUNT macro 1669
Mouse Control 778

N

NIC_TYPE_BRIDGE enumeration member 1013
NIC_TYPE_PHYSICAL enumeration member 1013
NIC_TYPE_VETH enumeration member 1013
NIC_TYPE_VLAN enumeration member 1013
NRM_DISABLED enumeration member 1014
NRM_ENABLED enumeration member 1014

O

Organization of This Guide 2
Other Macros 1654
Other PRL Macros 1620
Other PRL Structs 953
Other PRL Types 809
Other Structs 1102
Other Types 835

P

P3D_DISABLED enumeration member 1032
P3D_ENABLED_DX9 enumeration member 1032
P3D_ENABLED_HIGHEST enumeration member 1032
PACF_CANCEL_TASK_ON_END_SESSION enumeration member 966
PACF_HIGH_SECURITY enumeration member 966
PACF_LAST enumeration member 966
PACF_MAX enumeration member 966
PACF_NON_INTERACTIVE_MODE enumeration member 966
PACF_NORMAL_SECURITY enumeration member 966
PADS_CANCELED enumeration member 972
PADS_DOWNLOADED enumeration member 972
PADS_DOWNLOADING enumeration member 972
PADS_MOVED enumeration member 972
PADS_NOT_STARTED enumeration member 972
PADS_REGISTERED enumeration member 972
PADS_STOPPED enumeration member 972
PADS_UNPACKED enumeration member 972
PAI_GENERATE_INDEX enumeration member 965
PAI_INVALID_ADAPTER enumeration member 965
PAIF_INIT_AS_APPSTORE_CLIENT enumeration member 967
PAIF_USE_GRAPHIC_MODE enumeration member 967
PAM_LAST enumeration member 972
PAM_SERVER enumeration member 972
PAM_UNKNOWN enumeration member 972
PAO_VM_NOT_SHARED enumeration member 923
PAO_VM_SHARED_ON_FULL_ACCESS enumeration member 923
PAO_VM_SHARED_ON_VIEW enumeration member 923
PAO_VM_SHARED_ON_VIEW_AND_RUN enumeration member 923
PAO_VM_SHUTDOWN enumeration member 925
PAO_VM_START_MANUAL enumeration member 925
PAO_VM_START_ON_GUI_APP_STARTUP enumeration member 925

PAO_VM_START_ON_GUI_VM_WINDOW_OPEN enumeration member 925

PAO_VM_START_ON_GUI_WINDOW_LOAD enumeration member 925

PAO_VM_START_ON_LOAD enumeration member 925

PAO_VM_START_ON_RELOAD enumeration member 925

PAO_VM_STOP enumeration member 925

PAO_VM_SUSPEND enumeration member 925

PAR_MAX enumeration member 965

PAR_SRV_SENDPROBLEMREPORT_ACCESS enumeration member 965

PAR_SRV_SERVER_PROFILE_BEGINEDIT_ACCESS enumeration member 965

PAR_SRV_SERVER_PROFILE_COMMIT_ACCESS enumeration member 965

PAR_SRV_USER_PROFILE_BEGINEDIT_ACCESS enumeration member 965

PAR_SRV_USER_PROFILE_COMMIT_ACCESS enumeration member 965

PAR_VM_AUTHORISE_ACCESS enumeration member 965

PAR_VM_BEGINEDIT_ACCESS enumeration member 965

PAR_VM_CANCEL_COMPACT_ACCESS enumeration member 965

PAR_VM_CANCEL_COMPRESSOR_ACCESS enumeration member 965

PAR_VM_CHANGE_GUEST_OS_PASSWORD_ACCESS enumeration member 965

PAR_VM_CHANGE_PASSWORD_ACCESS enumeration member 965

PAR_VM_CHANGE_SID_ACCESS enumeration member 965

PAR_VM_CLONE_ACCESS enumeration member 965

PAR_VM_COMMIT_ACCESS enumeration member 965

PAR_VM_COMPACT_ACCESS enumeration member 965

PAR_VM_CONVERT_DISKS_ACCESS enumeration member 965

PAR_VM_CREATE_ACCESS enumeration member 965

PAR_VM_CREATE_BACKUP_ACCESS enumeration member 965

PAR_VM_CREATE_SNAPSHOT_ACCESS enumeration member 965

PAR_VM_DECRYPT_ACCESS enumeration member 965

PAR_VM_DELETE_ACCESS enumeration member 965

PAR_VM_DELETE_SNAPSHOT_ACCESS enumeration member 965

PAR_VM_DROPSUSPENDEDSTATE_ACCESS enumeration member 965

PAR_VM_EDITING_ACCESS enumeration member 965

PAR_VM_ENCRYPT_ACCESS enumeration member 965

PAR_VM_ENCRYPT_OPERATION_ACCESS enumeration member 965

PAR_VM_GET_SUSPENDED_SCREEN_ACCESS enumeration member 965

PAR_VM_GET_TOOLS_INFO enumeration member 965

PAR_VM_GET_VIRT_DEV_INFO enumeration member 965

PAR_VM_GET_VMINFO_ACCESS enumeration member 965

PAR_VM_GETCONFIG_ACCESS enumeration member 965

PAR_VM_GETPROBLEMREPORT_ACCESS enumeration member 965

PAR_VM_GETSTATISTICS_ACCESS enumeration member 965

PAR_VM_GUI_VIEW_MODE_CHANGE_ACCESS enumeration member 965

PAR_VM_INITIATE_DEV_STATE_NOTIFICATIONS_ACCESS enumeration member 965

PAR_VM_INSTALL_TOOLS_ACCESS enumeration member 965

PAR_VM_INSTALL_UTILITY_ACCESS enumeration member 965

PAR_VM_INTERNAL_CMD_ACCESS enumeration member 965

PAR_VM_LOCK_ACCESS enumeration member 965

PAR_VM_MIGRATE_ACCESS enumeration member 965

PAR_VM_MIGRATE_CANCEL_ACCESS enumeration member 965

PAR_VM_MOUNT_ACCESS enumeration member 965

PAR_VM_MOVE_ACCESS enumeration member 965

PAR_VM_PAUSE_ACCESS enumeration member 965

PAR_VM_PERFSTAT_ACCESS enumeration member 965

PAR_VM_REGISTER_ACCESS enumeration member 965

PAR_VM_REMOVE_PROTECTION_ACCESS enumeration member 965

PAR_VM_RESET_ACCESS enumeration member 965

PAR_VM_RESET_UPTIME_ACCESS enumeration member 965

PAR_VM_RESIZE_DISK_ACCESS enumeration member 965

PAR_VM_RESTART_GUEST_ACCESS enumeration member 965

PAR_VM_RESTORE_ACCESS enumeration member 965

PAR_VM_RESTORE_BACKUP_ACCESS enumeration

member 965
PAR_VM_RESUME_ACCESS enumeration member 965
PAR_VM_RUN_COMPRESSOR_ACCESS enumeration member 965
PAR_VM_SEND_ANSWER_ACCESS enumeration member 965
PAR_VM_SET_PROTECTION_ACCESS enumeration member 965
PAR_VM_START_ACCESS enumeration member 965
PAR_VM_START_EX_ACCESS enumeration member 965
PAR_VM_START_STOP_VNC_SERVER enumeration member 965
PAR_VM_STATISTICS_SUBSCRIPTION_ACCESS enumeration member 965
PAR_VM_STOP_ACCESS enumeration member 965
PAR_VM_SUBSCRIBETOGUESTSTATISTICS_ACCESS enumeration member 965
PAR_VM_SUSPEND_ACCESS enumeration member 965
PAR_VM_SWITCH_TO_SNAPSHOT_ACCESS enumeration member 965
PAR_VM_UNLOCK_ACCESS enumeration member 965
PAR_VM_UNREG_ACCESS enumeration member 965
PAR_VM_UNSUBSCRIBEFROMGUESTSTATISTICS_ACCESS enumeration member 965
PAR_VM_UPDATE_SECURITY_ACCESS enumeration member 965
PAR_VM_UPDATE_TOOLS_SECTION enumeration member 965
PAR_VMDEV_CONNECT_ACCESS enumeration member 965
PAR_VMDEV_COPY_IMAGE_ACCESS enumeration member 965
PAR_VMDEV_CREATEIMAGE_ACCESS enumeration member 965
PAR_VMDEV_DISCONNECT_ACCESS enumeration member 965
PARALLELS_API_VER 1669
PARALLELS_API_VER macro 1669
PARALLELS_API_VER_1 1669
PARALLELS_API_VER_1 macro 1669
PARALLELS_API_VER_2 1670
PARALLELS_API_VER_2 macro 1670
PARALLELS_API_VER_3 1670
PARALLELS_API_VER_3 macro 1670
PARALLELS_API_VER_4 1670
PARALLELS_API_VER_4 macro 1670
PARALLELS_API_VER_5 1670
PARALLELS_API_VER_5 macro 1670
PARALLELS_API_VER_6 1670
PARALLELS_API_VER_6 macro 1670
PARALLELS_API_VER_7 1671
PARALLELS_API_VER_7 macro 1671
PARALLELS_SDK_LIB_MAJ 1671
PARALLELS_SDK_LIB_MAJ macro 1671
PAS_CLOSE_VM_WINDOW enumeration member 924
PAS_KEEP_VM_WINDOW_OPEN enumeration member 924
PAS_QUIT_APPLICATION enumeration member 924
PBMBF_CREATE_MAP enumeration member 972
PBO_CD_HDD_FLOPPY enumeration member 927
PBO_FLOPPY_HDD_CD enumeration member 927
PBO_HDD_CD_FLOPPY enumeration member 927
PBSL_HIGH_SECURITY enumeration member 925
PBSL_LOW_SECURITY enumeration member 925
PBSL_NORMAL_SECURITY enumeration member 925
PBT_BACKUP_ID enumeration member 925
PBT_CHAIN enumeration member 925
PBT_CT enumeration member 925
PBT_DIFFERENTIAL enumeration member 925
PBT_DIRECT_DATA_CONNECTION enumeration member 925
PBT_EFI enumeration member 973
PBT_FULL enumeration member 925
PBT_IGNORE_NOT_EXISTS enumeration member 925
PBT_INCREMENTAL enumeration member 925
PBT_KEEP_CHAIN enumeration member 925
PBT_LEGACY enumeration member 973
PBT_RESTORE_TO_COPY enumeration member 925
PBT_UNCOMPRESSED enumeration member 925
PBT_VM enumeration member 925
PCA_COHERENCE enumeration member 976
PCA_CRYSTAL enumeration member 976
PCA_HIDE_APPLICATION enumeration member 976
PCA_MODALITY enumeration member 976
PCA_NO_ACTION enumeration member 976

PCA_SEAMLESS enumeration member 976
PCA_UNGRAB_INPUT enumeration member 976
PCA_WINDOWED enumeration member 976
PCC_AUDIT_WRITE enumeration member 979
PCC_CHOWN enumeration member 979
PCC_DAC_OVERRIDE enumeration member 979
PCC_DAC_READ_SEARCH enumeration member 979
PCC_FOWNER enumeration member 979
PCC_FS_MASK enumeration member 979
PCC_FSETID enumeration member 979
PCC_IPC_LOCK enumeration member 979
PCC_IPC_OWNER enumeration member 979
PCC_KILL enumeration member 979
PCC_LEASE enumeration member 979
PCC_LINUX_IMMUTABLE enumeration member 979
PCC_MKNOD enumeration member 979
PCC_NET_ADMIN enumeration member 979
PCC_NET_BIND_SERVICE enumeration member 979
PCC_NET_BROADCAST enumeration member 979
PCC_NET_RAW enumeration member 979
PCC_SETFCAP enumeration member 979
PCC_SETGID enumeration member 979
PCC_SETPCAP enumeration member 979
PCC_SETUID enumeration member 979
PCC_SYS_ADMIN enumeration member 979
PCC_SYS_BOOT enumeration member 979
PCC_SYS_CHROOT enumeration member 979
PCC_SYS_MODULE enumeration member 979
PCC_SYS_NICE enumeration member 979
PCC_SYS_PACCT enumeration member 979
PCC_SYS_PTRACE enumeration member 979
PCC_SYS_RAWIO enumeration member 979
PCC_SYS_RESOURCE enumeration member 979
PCC_SYS_TIME enumeration member 979
PCC_SYS_TTY_CONFIG enumeration member 979
PCC_VE_ADMIN enumeration member 979
PCD_BUSLOGIC enumeration member 974
PCD_LSI_SAS enumeration member 974
PCD_LSI_SPI enumeration member 974
PCD_UNKNOWN_DEVICE enumeration member 974
PCD_VIRTIO_SCSI enumeration member 974
PCDIF_CREATE_FROM_LION_RECOVERY_PARTITION enumeration member 978
PCEF_CHANGE_MASTER_KEY enumeration member 974
PCEF_DONT_SHRED_PLAIN_DATA enumeration member 974
PCF_FEATURE_BRIDGE enumeration member 980
PCF_FEATURE_DEF_PERM enumeration member 980
PCF_FEATURE_IPGRE enumeration member 980
PCF_FEATURE_IPIP enumeration member 980
PCF_FEATURE_NFS enumeration member 980
PCF_FEATURE_NFSD enumeration member 980
PCF_FEATURE_PPP enumeration member 980
PCF_FEATURE_SIT enumeration member 980
PCF_FEATURE_SYSFS enumeration member 980
PCF_LIGHTWEIGHT_CLIENT enumeration member 837
PCF_ORIGINAL_CLIENT enumeration member 837
PCF_TRUSTED_CHANNEL enumeration member 837
PCFE_EXT_00000007_EBX enumeration member 976
PCFE_EXT_0000000D_EAX enumeration member 976
PCFE_EXT_80000001_ECX enumeration member 976
PCFE_EXT_80000001_EDX enumeration member 976
PCFE_EXT_80000007_EDX enumeration member 976
PCFE_EXT_80000008_EAX enumeration member 976
PCFE_EXT_FEATURES enumeration member 976
PCFE_FEATURES enumeration member 976
PCFE_MAX enumeration member 976
PCM_COMPACT_WITH_HARD_DISKS_INFO enumeration member 974
PCM_CPU_AMD_V enumeration member 976
PCM_CPU_INTEL_VT_X enumeration member 976
PCM_CPU_MODE_32 enumeration member 977
PCM_CPU_MODE_64 enumeration member 977
PCM_CPU_NONE_HV enumeration member 976
PCM_FULL_CLEAN_UP_VM enumeration member 974
PCM_HARD_DISKS_INFO enumeration member 974
PCNM_DISABLED enumeration member 1014
PCNM_FULL enumeration member 1014
PCNM_NOT_SET enumeration member 1014
PCNM_STATEFUL enumeration member 1014

PCNM_STATELESS enumeration member 1014
PCR_DCACHESIZE enumeration member 980
PCR_DGRAMRCVBUF enumeration member 980
PCR_KMEMSIZE enumeration member 980
PCR_LAST enumeration member 980
PCR_LOCKEDPAGES enumeration member 980
PCR_NUMFILE enumeration member 980
PCR_NUMFLOCK enumeration member 980
PCR_NUMIPTENT enumeration member 980
PCR_NUMOTHERSOCK enumeration member 980
PCR_NUMPROC enumeration member 980
PCR_NUMPTY enumeration member 980
PCR_NUMSIGINFO enumeration member 980
PCR_NUMTCPSOCK enumeration member 980
PCR_OOMGUARPAGES enumeration member 980
PCR_OTHERSOCKBUF enumeration member 980
PCR_PHYSPAGES enumeration member 980
PCR_PRIVVMPAGES enumeration member 980
PCR_QUOTAUGIDLIMIT enumeration member 980
PCR_SHMPAGES enumeration member 980
PCR_SWAPPAGES enumeration member 980
PCR_TCPRCVBUF enumeration member 980
PCR_TCPSNDBUF enumeration member 980
PCR_VMGUARPAGES enumeration member 980
PCS_CONNECTED enumeration member 1021
PCS_CONNECTING enumeration member 1021
PCS_DISCONNECTED enumeration member 1021
PCSF_BACKUP enumeration member 978
PCSF_DISK_ONLY enumeration member 978
PCT_TYPE_EZ_APP enumeration member 981
PCT_TYPE_EZ_OS enumeration member 981
PCTMP_HIGH_SECURITY enumeration member 975
PCTMP_LOW_SECURITY enumeration member 975
PCTMP_NORMAL_SECURITY enumeration member 975
PCTMPL_FORCE enumeration member 975
PCVD_CANCEL enumeration member 975
PCVD_MERGE_ALL_SNAPSHOTS enumeration member 975
PCVD_TO_EXPANDING_DISK enumeration member 975
PCVD_TO_NON_SPLIT_DISK enumeration member 975
PCVD_TO_PLAIN_DISK enumeration member 975
PCVD_TO_SPLIT_DISK enumeration member 975
PCVF_CHANGE_SID enumeration member 973
PCVF_CLONE_TO_TEMPLATE enumeration member 973
PCVF_DETACH_EXTERNAL_VIRTUAL_HDD enumeration member 973
PCVF_IMPORT_BOOT_CAMP enumeration member 973
PCVF_LINKED_CLONE enumeration member 973
PDA_DEVICE_IDLE enumeration member 982
PDA_DEVICE_READ enumeration member 982
PDA_DEVICE_WRITE enumeration member 982
PDBF_BGR32 enumeration member 983
PDBF_INDEX8 enumeration member 983
PDBF_NO_CONVERSION enumeration member 983
PDBF_RGB15 enumeration member 983
PDBF_RGB16 enumeration member 983
PDBF_RGB24 enumeration member 983
PDBF_RGB32 enumeration member 983
PDBF_YUV420 enumeration member 983
PDBF_YUV422x4 enumeration member 983
PDBF_YUV422x6 enumeration member 983
PDBF_YUV422x8 enumeration member 983
PDCC_HIGH_COMPRESSION enumeration member 983
PDCC_LOW_COMPRESSION enumeration member 983
PDCC_MEDIUM_COMPRESSION enumeration member 983
PDCC_NO_COMPRESSION enumeration member 983
PDCM_CLIENT_DISPLAY_INFO enumeration member 984
PDCM_DISABLE_HIDPI_ADJUSTMENT enumeration member 984
PDCM_DO_NOT_RESTORE enumeration member 984
PDCM_ENABLE_HEADS enumeration member 984
PDCM_HAS_HIDPI_DISPLAY enumeration member 984
PDCM_RESTORE_DEFAULT enumeration member 984
PDCM_RETINA_MODE enumeration member 984
PDCQ_HIGH_QUALITY enumeration member 984
PDCQ_LOW_QUALITY enumeration member 984
PDCQ_MEDIUM_QUALITY enumeration member 984
PDCT_HIGH_QUALITY_WITH_COMPRESSION enumeration member 984
PDCT_HIGH_QUALITY_WITHOUT_COMPRESSION enumeration member 984

PDCT_LOW_QUALITY_WITH_COMPRESSION enumeration member 984

PDCT_LOW_QUALITY_WITHOUT_COMPRESSION enumeration member 984

PDCT_MEDIUM_QUALITY_WITH_COMPRESSION enumeration member 984

PDCT_MEDIUM_QUALITY_WITHOUT_COMPRESSION enumeration member 984

PDE_ATTACHED_BACKUP_DISK enumeration member 982

PDE_CLUSTERED_DEVICE enumeration member 982

PDE_FLOPPY_DISK enumeration member 982

PDE_GENERIC_DEVICE enumeration member 982

PDE_GENERIC_NETWORK_ADAPTER enumeration member 982

PDE_GENERIC_PCI_DEVICE enumeration member 982

PDE_GENERIC_PORT enumeration member 982

PDE_GENERIC_SCSI_DEVICE enumeration member 982

PDE_HARD_DISK enumeration member 982

PDE_MASSSTORAGE_DEVICE enumeration member 982

PDE_MAX enumeration member 982

PDE_MIXER_DEVICE enumeration member 982

PDE_OPTICAL_DISK enumeration member 982

PDE_PARALLEL_PORT enumeration member 982

PDE_PCI_VIDEO_ADAPTER enumeration member 982

PDE_PRINTER enumeration member 982

PDE_SERIAL_PORT enumeration member 982

PDE_SOUND_DEVICE enumeration member 982

PDE_STORAGE_DEVICE enumeration member 982

PDE_USB_DEVICE enumeration member 982

PDE_VIRTUAL_SHARED_FOLDERS_DEVICE enumeration member 982

PDE_VIRTUAL_SNAPSHOT_DEVICE enumeration member 982

PDM_APP_IN_DOCK_ALWAYS enumeration member 971

PDM_APP_IN_DOCK_COHERENCE_ONLY enumeration member 971

PDM_APP_IN_DOCK_NEVER enumeration member 971

PDR_ANOTHER_CLIENT_CONNECTED enumeration member 838

PDR_HOST_USER_INPUT_DETECTED enumeration member 838

PDR_HOST_USER_LOGGED_IN enumeration member 838

PDR_RECONNECT enumeration member 838

PDR_SESSION_BECOME_INACTIVE enumeration member 838

PDR_UNSPECIFIED enumeration member 838

PDSF_BACKUP enumeration member 981

PDSF_BACKUP_MAP enumeration member 981

PDSF_SKIP_SPACE_CHECK enumeration member 981

PDSS_UNKNOWN enumeration member 981

PDSS_VM_DISK_DATA_SPACE enumeration member 981

PDSS_VM_FULL_SPACE enumeration member 981

PDSS_VM_MISCELLANEOUS_SPACE enumeration member 981

PDSS_VM_RECLAIM_SPACE enumeration member 981

PDSS_VM_SNAPSHOTS_SPACE enumeration member 981

PDT_ANY_TYPE enumeration member 929

PDT_USE_AC97_SOUND enumeration member 929

PDT_USE_BOOTCAMP enumeration member 929

PDT_USE_BRIDGE_ETHERNET enumeration member 929

PDT_USE_CREATIVE_SB16_SOUND enumeration member 929

PDT_USE_DIRECT_ASSIGN enumeration member 929

PDT_USE_FILE_SYSTEM enumeration member 929

PDT_USE_HOST_ONLY_NETWORK enumeration member 929

PDT_USE_IMAGE_FILE enumeration member 929

PDT_USE_OTHER enumeration member 929

PDT_USE_OUTPUT_FILE enumeration member 929

PDT_USE_PARALLEL_PORT_PRINT_TO_PDF_MODE enumeration member 929

PDT_USE_PARALLEL_PORT_PRINTER_MODE enumeration member 929

PDT_USE_REAL_CDDVD enumeration member 929

PDT_USE_REAL_DEVICE enumeration member 929

PDT_USE_REAL_FLOPPY enumeration member 929

PDT_USE_REAL_HDD enumeration member 929

PDT_USE_REAL_PARALLEL_PORT enumeration member 929

PDT_USE_REAL_SERIAL_PORT enumeration member 929

PDT_USE_REAL_USB_CONTROLLER enumeration member 929

PDT_USE_SERIAL_PORT_SOCKET_MODE enumeration member 929

PDT_USE_SERIAL_PORT_TCP_MODE enumeration member 929

PDT_USE_SERIAL_PORT_UDP_MODE enumeration member

929
PDT_USE_SHARED_NETWORK enumeration member 929
PDT_USE_TCP enumeration member 929
PDT_USE_UDP enumeration member 929
PEMBF_FAILURE_CLEANUP enumeration member 986
PET Macros 1106
PET_ANSWER_APPEND 1108
PET_ANSWER_APPEND macro 1108
PET_ANSWER_BREAK 1109
PET_ANSWER_BREAK macro 1109
PET_ANSWER_BROWSE 1109
PET_ANSWER_BROWSE macro 1109
PET_ANSWER_CANCEL 1109
PET_ANSWER_CANCEL macro 1109
PET_ANSWER_CHANGE 1109
PET_ANSWER_CHANGE macro 1109
PET_ANSWER_COMMIT 1110
PET_ANSWER_COMMIT macro 1110
PET_ANSWER_COMPACT 1110
PET_ANSWER_COMPACT macro 1110
PET_ANSWER_CONTINUE 1110
PET_ANSWER_CONTINUE macro 1110
PET_ANSWER_COPIED 1110
PET_ANSWER_COPIED macro 1110
PET_ANSWER_CREATE 1110
PET_ANSWER_CREATE macro 1110
PET_ANSWER_CREATE_NEW 1111
PET_ANSWER_CREATE_NEW macro 1111
PET_ANSWER_DISCONNECT_ANYWAY 1111
PET_ANSWER_DISCONNECT_ANYWAY macro 1111
PET_ANSWER_DONT_CHANGE 1111
PET_ANSWER_DONT_CHANGE macro 1111
PET_ANSWER_LATER 1111
PET_ANSWER_LATER macro 1111
PET_ANSWER_MOVED 1111
PET_ANSWER_MOVED macro 1111
PET_ANSWER_NO 1112
PET_ANSWER_NO macro 1112
PET_ANSWER_OK 1112
PET_ANSWER_OK macro 1112
PET_ANSWER_OVERIDE 1112
PET_ANSWER_OVERIDE macro 1112
PET_ANSWER_REPLACE 1112
PET_ANSWER_REPLACE macro 1112
PET_ANSWER_RESTART 1112
PET_ANSWER_RESTART macro 1112
PET_ANSWER_RESTART_NOW 1113
PET_ANSWER_RESTART_NOW macro 1113
PET_ANSWER_RESTORE 1113
PET_ANSWER_RESTORE macro 1113
PET_ANSWER_RESUME 1113
PET_ANSWER_RESUME macro 1113
PET_ANSWER_RETRY 1113
PET_ANSWER_RETRY macro 1113
PET_ANSWER_REVERT 1114
PET_ANSWER_REVERT macro 1114
PET_ANSWER_SHUTDOWN 1114
PET_ANSWER_SHUTDOWN macro 1114
PET_ANSWER_SKIP 1114
PET_ANSWER_SKIP macro 1114
PET_ANSWER_STARTVM 1114
PET_ANSWER_STARTVM macro 1114
PET_ANSWER_STOP 1114
PET_ANSWER_STOP macro 1114
PET_ANSWER_STOP_AND_RESTORE 1115
PET_ANSWER_STOP_AND_RESTORE macro 1115
PET_ANSWER_SUSPEND 1115
PET_ANSWER_SUSPEND macro 1115
PET_ANSWER_SUSPEND_ANYWAY 1115
PET_ANSWER_SUSPEND_ANYWAY macro 1115
PET_ANSWER_USE_CURRENT 1115
PET_ANSWER_USE_CURRENT macro 1115
PET_ANSWER_YES 1115
PET_ANSWER_YES macro 1115
PET_DESKCTL_ACCEPT_CLIENTS enumeration member 1016
PET_DESKCTL_ATTACH enumeration member 1016
PET_DESKCTL_ATTACH_FIN enumeration member 1016
PET_DESKCTL_CONFIG enumeration member 1016

PET_DESKCTL_CONFIG_QUERY enumeration member 1016

PET_DESKCTL_LOCAL_AUTH enumeration member 1016

PET_DESKCTL_LOCAL_AUTH_ANSWER enumeration member 1016

PET_DESKCTL_LOCAL_AUTH_FIN enumeration member 1016

PET_DESKCTL_LOCAL_AUTH_QUEST enumeration member 1016

PET_DESKCTL_REJECT_CLIENTS enumeration member 1016

PET_DESKCTL_SEND_SAS enumeration member 1016

PET_DESKCTL_SESSION_AUTH enumeration member 1016

PET_DESKCTL_SESSION_AUTH_FIN enumeration member 1016

PET_DESKCTL_STATUS_CHANGE enumeration member 1016

PET_DSP_COHERENCE_FAKE_STUB_NOTIFICATION enumeration member 986

PET_DSP_EVT_ANSWER_TO_VM_WAS_DONE enumeration member 986

PET_DSP_EVT_APPLIANCE_ARCHIVE_UNPACK_FINISHED enumeration member 986

PET_DSP_EVT_APPLIANCE_ARCHIVE_UNPACK_STARTED enumeration member 986

PET_DSP_EVT_APPLIANCE_DOWNLOAD_FINISHED enumeration member 986

PET_DSP_EVT_APPLIANCE_DOWNLOAD_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_APPLIANCE_REGISTER_VM_FINISHED enumeration member 986

PET_DSP_EVT_APPLIANCE_REGISTER_VM_STARTED enumeration member 986

PET_DSP_EVT_BACKUP_CANCELLED enumeration member 986

PET_DSP_EVT_BACKUP_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_BACKUP_STARTED enumeration member 986

PET_DSP_EVT_CEP_WRITE_STRING enumeration member 986

PET_DSP_EVT_CLIENT_CONNECTED enumeration member 986

PET_DSP_EVT_CLIENT_DISCONNECTED enumeration member 986

PET_DSP_EVT_CLIENT_REJECTED enumeration member 986

PET_DSP_EVT_COMMIT_UNFINISHED_DISK_OP_FINISHED enumeration member 986

PET_DSP_EVT_COMMIT_UNFINISHED_DISK_OP_STARTED enumeration member 986

PET_DSP_EVT_COMMON_PREFS_CHANGED enumeration member 986

PET_DSP_EVT_CONNECTED_TO_PROXY enumeration member 986

PET_DSP_EVT_CONNECTING_TO_PROXY enumeration member 986

PET_DSP_EVT_CONNECTION_CLIENT_INFO_WAS_SET enumeration member 986

PET_DSP_EVT_CONNECTION_WAS_CLOSED enumeration member 986

PET_DSP_EVT_CONVERSION_DISKS_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_CONVERSION_DISKS_PROGRESS_FINISHED enumeration member 986

PET_DSP_EVT_CONVERT_THIRD_PARTY_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_CREATE_BACKUP_FINISHED enumeration member 986

PET_DSP_EVT_DIR_STATE_CHANGED enumeration member 986

PET_DSP_EVT_DISCONNECTED_FROM_PROXY enumeration member 986

PET_DSP_EVT_DISK_RESIZE_FINISHED enumeration member 986

PET_DSP_EVT_DISK_RESIZE_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_DISK_RESIZE_STARTED enumeration member 986

PET_DSP_EVT_DISP_CONNECTION_CLOSED enumeration member 986

PET_DSP_EVT_DISP_SHUTDOWN enumeration member 986

PET_DSP_EVT_ENCRYPTED_VM_NEED_AUTH_AGAIN enumeration member 986

PET_DSP_EVT_ENCRYPTED_VM_PASSWORD_CHANGED enumeration member 986

PET_DSP_EVT_ERROR_MESSAGE enumeration member 986

PET_DSP_EVT_FOUND_LOST_VM_CONFIG enumeration member 986

PET_DSP_EVT_HAS_EVENT enumeration member 986

PET_DSP_EVT_HOST_STATISTICS_UPDATED enumeration

member 986

PET_DSP_EVT_HW_CONFIG_CHANGED enumeration member 986

PET_DSP_EVT_JOB_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_LICENSE_CHANGED enumeration member 986

PET_DSP_EVT_OP_ACCEPTED enumeration member 986

PET_DSP_EVT_OP_REJECTED enumeration member 986

PET_DSP_EVT_OP_RESULT enumeration member 986

PET_DSP_EVT_PERFSTATS enumeration member 986

PET_DSP_EVT_REBOOT_HOST enumeration member 986

PET_DSP_EVT_REBOOT_HOST_MANUAL_TO_INIT_VTD_DEVICES
enumeration member 986

PET_DSP_EVT_RECONFIG_VM_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_RECONNECT enumeration member 986

PET_DSP_EVT_REMOVE_BACKUP_FINISHED enumeration member 986

PET_DSP_EVT_RESTORE_BACKUP_FINISHED enumeration member 986

PET_DSP_EVT_RESTORE_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_SMC_CANCEL_USER_COMMAND enumeration member 986

PET_DSP_EVT_SMC_DISPATCHER_SHUTDOWN enumeration member 986

PET_DSP_EVT_SMC_USER_FORCE_DISCONNECTED enumeration member 986

PET_DSP_EVT_TIME_MACHINE_BACKUP_IS_OVER enumeration member 986

PET_DSP_EVT_USER_PROFILE_CHANGED enumeration member 986

PET_DSP_EVT_VM_ABORTED enumeration member 986

PET_DSP_EVT_VM_ABOUT_TO_RESET enumeration member 986

PET_DSP_EVT_VM_ABOUT_TO_START_DEINIT enumeration member 986

PET_DSP_EVT_VM_ABOUT_TO_START_INIT enumeration member 986

PET_DSP_EVT_VM_ADDED enumeration member 986

PET_DSP_EVT_VM_ADDITION_STATE_CHANGED enumeration member 986

PET_DSP_EVT_VM_COMPACT_FINISHED enumeration member 986

PET_DSP_EVT_VM_COMPACT_PROCESSING enumeration member 986

PET_DSP_EVT_VM_COMPRESSOR_CONTINUED enumeration member 986

PET_DSP_EVT_VM_COMPRESSOR_FINISHED enumeration member 986

PET_DSP_EVT_VM_CONFIG_APPLIED enumeration member 986

PET_DSP_EVT_VM_CONFIG_APPLY_FINISHED enumeration member 986

PET_DSP_EVT_VM_CONFIG_CHANGED enumeration member 986

PET_DSP_EVT_VM_CONTINUED enumeration member 986

PET_DSP_EVT_VM_CONTINUED_BY_HOST_WAKEUP enumeration member 986

PET_DSP_EVT_VM_CREATED enumeration member 986

PET_DSP_EVT_VM_DECRYPT_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_VM_DELETED enumeration member 986

PET_DSP_EVT_VM_DELETING_STATE enumeration member 986

PET_DSP_EVT_VM_DEV_STATE_CHANGED enumeration member 986

PET_DSP_EVT_VM_ENCRYPT_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_VM_FROZEN enumeration member 986

PET_DSP_EVT_VM_IO_CLIENT_STATISTICTS enumeration member 986

PET_DSP_EVT_VM_LOW_HOST_MEMORY enumeration member 986

PET_DSP_EVT_VM_MAC_OS_VER enumeration member 986

PET_DSP_EVT_VM_MEMORY_SWAPPING_FINISHED enumeration member 986

PET_DSP_EVT_VM_MEMORY_SWAPPING_STARTED enumeration member 986

PET_DSP_EVT_VM_MESSAGE enumeration member 986

PET_DSP_EVT_VM_MIGRATE_CANCELLED enumeration member 986

PET_DSP_EVT_VM_MIGRATE_CANCELLED_DISP enumeration member 986

PET_DSP_EVT_VM_MIGRATE_FINISHED enumeration member 986

PET_DSP_EVT_VM_MIGRATE_FINISHED_DISP enumeration member 986

PET_DSP_EVT_VM_MIGRATE_PROGRESS_CHANGED enumeration member 986

PET_DSP_EVT_VM_MIGRATE_STARTED enumeration member 986

PET_DSP_EVT_VM_MIGRATE_WAIT_REMOUNT enumeration member 986

PET_DSP_EVT_VM_NET_RECONNECTED enumeration member 986

PET_DSP_EVT_VM_PAUSED enumeration member 986

PET_DSP_EVT_VM_PAUSED_BY_HOST_SLEEP enumeration member 986

PET_DSP_EVT_VM_PERFSTATS enumeration member 986

PET_DSP_EVT_VM_PIS_NOTIFICATION_ASK_TO_INSTALL enumeration member 986

PET_DSP_EVT_VM_PROBLEM_REPORT_CREATED enumeration member 986

PET_DSP_EVT_VM_QUESTION enumeration member 986

PET_DSP_EVT_VM_REMOTE_DEV_READY_TO_START enumeration member 986

PET_DSP_EVT_VM_RESETED enumeration member 986

PET_DSP_EVT_VM_RESTORED enumeration member 986

PET_DSP_EVT_VM_RESTORING enumeration member 986

PET_DSP_EVT_VM_RESUMED enumeration member 986

PET_DSP_EVT_VM_RESUMING enumeration member 986

PET_DSP_EVT_VM_SECURITY_CHANGED enumeration member 986

PET_DSP_EVT_VM_SET_HOST_TIME enumeration member 986

PET_DSP_EVT_VM_SNAPSHOTED enumeration member 986

PET_DSP_EVT_VM_SNAPSHOTING enumeration member 986

PET_DSP_EVT_VM_SNAPSHOTS_TREE_CHANGED enumeration member 986

PET_DSP_EVT_VM_SOFTWARE_INSTALLED enumeration member 986

PET_DSP_EVT_VM_START_IO_FLOW enumeration member 986

PET_DSP_EVT_VM_STARTED enumeration member 986

PET_DSP_EVT_VM_STATE_CHANGED enumeration member 986

PET_DSP_EVT_VM_STATE_DELETED enumeration member 986

PET_DSP_EVT_VM_STATERESTORED enumeration member 986

PET_DSP_EVT_VM_STATESAVED enumeration member 986

PET_DSP_EVT_VM_STATISTICS_UPDATED enumeration member 986

PET_DSP_EVT_VM_STOPPED enumeration member 986

PET_DSP_EVT_VM_SUSPENDED enumeration member 986

PET_DSP_EVT_VM_SUSPENDING enumeration member 986

PET_DSP_EVT_VM_TOOLS_STATE_CHANGED enumeration member 986

PET_DSP_EVT_VM_UNFROZEN enumeration member 986

PET_DSP_EVT_VM_UNREGISTERED enumeration member 986

PET_DSP_EVT_VM_UPGRADE_COMPLETED enumeration member 986

PET_DSP_EVT_VM_UPGRADE_INIT enumeration member 986

PET_DSP_EVT_VM_UPGRADE_INIT_PHY enumeration member 986

PET_DSP_EVT_VM_UPGRADE_INIT_TIMEOUT enumeration member 986

PET_DSP_EVT_VM_UPGRADE_INIT_VMW enumeration member 986

PET_DSP_EVT_VM_UPGRADE_STAGE_1 enumeration member 986

PET_DSP_EVT_VM_UPGRADE_STAGE_2 enumeration member 986

PET_DSP_EVT_VM_UPGRADE_STAGE_3 enumeration member 986

PET_DSP_EVT_VM_UPGRADE_UNKNOWN_ERROR enumeration member 986

PET_DSP_EVT_VM_VIRTUAL_DEVICES_STATE_CHANGED enumeration member 986

PET_DSP_EVT_VM_WAS_DECRYPTED enumeration member 986

PET_DSP_EVT_VM_WAS_ENCRYPTED enumeration member 986

PET_DSP_EVT_VM_WAS_PROTECTED enumeration member 986

PET_DSP_EVT_VM_WAS_UNPROTECTED enumeration member 986

PET_DSP_EVT_WARNING_MESSAGE enumeration member 986

PET_ERR_WARN_REACH_OVERCOMMIT_STATE_ON_SET MEM
1116

PET_ERR_WARN_REACH_OVERCOMMIT_STATE_ON_SET MEM
macro 1116

PET_IO_ATTACH_RESPONSE enumeration member 1016

PET_IO_AUDIO_OUTPUT_DATA enumeration member 986

PET_IO_AUDIO_OUTPUT_ENCODING_SET enumeration

member 986

PET_IO_AUDIO_OUTPUT_STOP enumeration member 986

PET_IO_AUDIO_OUTPUT_STREAM_STARTED enumeration member 986

PET_IO_AUTH_RESPONSE enumeration member 1016

PET_IO_AVAILABLE_DISPLAYS enumeration member 986

PET_IO_BEFORE_SCREEN_SIZE enumeration member 986

PET_IO_CLI_ATTACH_TO_VM enumeration member 1016

PET_IO_CLI_AUDIO_PLAYOUT_STREAM_START enumeration member 1016

PET_IO_CLI_AUDIO_PLAYOUT_STREAM_STOP enumeration member 1016

PET_IO_CLI_AUDIO_RECORDING_STREAM_START enumeration member 1016

PET_IO_CLI_AUDIO_RECORDING_STREAM_STOP enumeration member 1016

PET_IO_CLI_AUTHENTICATE_EXEC_SESSION enumeration member 1016

PET_IO_CLI_AUTHENTICATE_SESSION enumeration member 1016

PET_IO_CLI_CVSRC_CONNECT enumeration member 1016

PET_IO_CLI_CVSRC_DISCONNECT enumeration member 1016

PET_IO_CLI_DPI_SCREEN_SIZE enumeration member 1016

PET_IO_CLI_ENCODING_REQUEST enumeration member 1016

PET_IO_CLI_GRACEFUL_SHUTDOWN enumeration member 1016

PET_IO_CLI_KEYBOARD_KEY enumeration member 1016

PET_IO_CLI_KEYBOARD_SCANCODE enumeration member 1016

PET_IO_CLI_MOUSE_MOVE enumeration member 1016

PET_IO_CLI_MOUSE_NEED_CURSOR enumeration member 1016

PET_IO_CLI_MOUSE_SETPOS enumeration member 1016

PET_IO_CLI_REMOTE_COMMAND enumeration member 1016

PET_IO_CLI_SCREEN_CAPTURE_BUFFER enumeration member 1016

PET_IO_CLI_SCREEN_SET_SURFACE enumeration member 1016

PET_IO_CLI_SCREEN_SIZE enumeration member 1016

PET_IO_CLI_SCREEN_WATCH_UPDATES enumeration member 1016

PET_IO_CLI_SSO_LOCKED enumeration member 1016

PET_IO_CLI_SSO_SET_CREDENTIALS enumeration member 1016

PET_IO_CLI_SSO_SET_CREDENTIALS_ASYNC enumeration member 1016

PET_IO_CLI_SSO_UNLOCKED enumeration member 1016

PET_IO_CLI_SWITCH_SLIDING_MOUSE enumeration member 1016

PET_IO_CLI_TOOLS_ADJUST_WORKING_AREA enumeration member 1016

PET_IO_CLI_TOOLS_CLIPBOARD_DATA enumeration member 1016

PET_IO_CLI_TOOLS_EMPTY_RECYCLE_BIN enumeration member 1016

PET_IO_CLI_TOOLS_GENERAL_COMMAND enumeration member 1016

PET_IO_CLI_TOOLS_MOVE_TASKBAR enumeration member 1016

PET_IO_CLI_TOOLS_NOTIFY_COHERENCE_STATE enumeration member 1016

PET_IO_CLI_TOOLS_OPEN_FILE_BROWSER enumeration member 1016

PET_IO_CLI_TOOLS_SET_INDENTS enumeration member 1016

PET_IO_CLI_TOOLS_SET_TASKBAR_VISIBILITY enumeration member 1016

PET_IO_CLI_TOOLS_SHOW_RECYCLE_BIN enumeration member 1016

PET_IO_CLI_TOOLS_SIA_DATA enumeration member 1016

PET_IO_CLI_TOOLS_TIS_QUERY enumeration member 1016

PET_IO_CLI_TOOLS_TOGGLE_DESKTOP enumeration member 1016

PET_IO_CLI_TOOLS_UIEMU enumeration member 1016

PET_IO_CLI_TOOLS_UTILITY_COMMAND enumeration member 1016

PET_IO_CLI_TOOLS_VMCTG_COMMAND enumeration member 1016

PET_IO_CLIENT_PROCESSED_ALL_DESCS_DATA enumeration member 1016

PET_IO_CVSRC_CLOSE enumeration member 986

PET_IO_CVSRC_OPEN enumeration member 986

PET_IO_DEVICE_IS_IN_USE enumeration member 986

PET_IO_DISCONNECT_WITH_REASON enumeration member 1016

PET_IO_DISPLAY_BEFORE_SCREEN_SIZE enumeration member 986

PET_IO_DISPLAY_GAMMA_CHANGED enumeration member 986

PET_IO_DISPLAY_PALETTE_CHANGED enumeration member 1016

PET_IO_DISPLAY_RESOLUTION_SET enumeration member 1016

PET_IO_DISPLAY_SCREEN_CAPTURED enumeration member 986

PET_IO_DISPLAY_SCREEN_SIZE enumeration member 986

PET_IO_DYNRES_TOOL_STATUS enumeration member 986

PET_IO_ENCODING_RESPONSE enumeration member 1016

PET_IO_EVT_DISCONNECT_WITH_REASON enumeration member 986

PET_IO_FIN_TO_TRANSMIT_STDOUT_STDERR enumeration member 1016

PET_IO_IOSERVICE_CHANNELS_PROTOCOL enumeration member 1016

PET_IO_KEYBOARD_LEDS enumeration member 986

PET_IO_MEMORY_INFO enumeration member 1016

PET_IO_MOUSE_CURSOR_CHANGED enumeration member 986

PET_IO_MOUSE_CURSOR_HID enumeration member 986

PET_IO_MOUSE_CURSOR_MOVED enumeration member 986

PET_IO_MOUSE_CURSOR_SET enumeration member 986

PET_IO_PTM_PROTOCOL enumeration member 1016

PET_IO_READY_TO_ACCEPT_STDIN_PKGS enumeration member 986

PET_IO_READY_TO_ACCEPT_STDOUT_STDERR_PKGS enumeration member 1016

PET_IO_REMOTE_COMMAND enumeration member 986

PET_IO_SCREEN_CAPTURED enumeration member 986

PET_IO_SCREEN_CAPTURED_BUFFER enumeration member 1016

PET_IO_SCREEN_SIZE enumeration member 986

PET_IO_SCREEN_SURFACE_ATTACHED enumeration member 986

PET_IO_SCREEN_SURFACE_DETACHED enumeration member 986

PET_IO_SCREEN_UPDATED enumeration member 1016

PET_IO_SLIDING_MOUSE_FLAG enumeration member 986

PET_IO_SSO_GET_CREDENTIALS enumeration member 986

PET_IO_STATE enumeration member 986

PET_IO_STDERR_PORTION enumeration member 1016

PET_IO_STDIN_PORTION enumeration member 1016

PET_IO_STDIN_WAS_CLOSED enumeration member 1016

PET_IO_STDOUT_PORTION enumeration member 1016

PET_IO_SWITCH_VESA_MODE enumeration member 986

PET_IO_TOOLS_CLIPBOARD_DATA enumeration member 986

PET_IO_TOOLS_DESKTOP_UTILITY_STATE enumeration member 986

PET_IO_TOOLS_DRAGDROP_DATA enumeration member 986

PET_IO_TOOLS_GENERAL_COMMAND enumeration member 986

PET_IO_TOOLS_LANGUAGE_HOTKEY_CHANGED enumeration member 986

PET_IO_TOOLS_SIA_DATA enumeration member 986

PET_IO_TOOLS_TIS_QUERY enumeration member 1016

PET_IO_TOOLS_TIS_SUBSCRIBE enumeration member 1016

PET_IO_TOOLS_TIS_UNSUBSCRIBE enumeration member 1016

PET_IO_TOOLS_UTILITY_COMMAND enumeration member 986

PET_IO_TOOLS_VM_SHUTDOWN enumeration member 986

PET_IO_TOOLS_VMCTG_COMMAND enumeration member 986

PET_IO_TTY_SETTINGS_PROVIDED enumeration member 1016

PET_IO_TTY_SETTINGS_REQUIRED enumeration member 1016

PET_IO_UIEMU_CARET_INFO enumeration member 986

PET_IO_UIEMU_ELEMENT_AT_POS enumeration member 986

PET_IT_COMPACTING_START enumeration member 1016

PET_IT_CONVERT_TO_CURRENT_START enumeration member 1016

PET_IT_CONVERT_TO_EXTEND_START enumeration member 1016

PET_IT_CONVERT_TO_OLD_START enumeration member 1016

PET_IT_DECREASING_START enumeration member 1016

PET_IT_FIX_CONSISTENCY_START enumeration member 1016

PET_IT_GET_RESIZE_INFORMATION_START enumeration member 1016

PET_IT_INCREASE_LAST_VOLUME_START enumeration member 1016

PET_IT_MAKE_EXPANDING_START enumeration member 1016

PET_IT_MAKE_NON_SPLIT_START enumeration member 1016

PET_IT_MAKE_PLAIN_START enumeration member 1016

PET_IT_MAKE_SPLIT_START enumeration member 1016

PET_IT_MERGE_SNAPSHOTS_START enumeration member 1016

PET_IT_RECONFIG_BOOTCAMP_START enumeration member 1016

PET_IT_RECONFIG_START enumeration member 1016

PET_IT_RESIZING_START enumeration member 1016

PET_IT_VALIDATE_BOOTCAMP_START enumeration member 1016

PET_JOB_COMMIT_UNFINISHED_DISK_OP_PROGRESS_CHANGED enumeration member 986

PET_JOB_CREATE_SNAPSHOT_PROGRESS_CHANGED enumeration member 986

PET_JOB_DELETE_SNAPSHOT_PROGRESS_CHANGED enumeration member 986

PET_JOB_DELETE_VM_PROGRESS_CHANGED enumeration member 986

PET_JOB_FILE_COPY_PROGRESS_CHANGED enumeration member 986

PET_JOB_HDD_CREATE_PROGRESS_CHANGED enumeration member 986

PET_JOB_RESUME_PROGRESS_CHANGED enumeration member 986

PET_JOB_STAGE_PROGRESS_CHANGED enumeration member 986

PET_JOB_SUSPEND_PROGRESS_CHANGED enumeration member 986

PET_JOB_SWITCH_TO_SNAPSHOT_PROGRESS_CHANGED enumeration member 986

PET_PTM_EVT_OBJECT_CREATED enumeration member 986

PET_PTM_EVT_OBJECT_DELETED enumeration member 986

PET_PTM_EVT_PROPERTY_CHANGED enumeration member 986

PET_PTM_EVT_RPC_COMPLETED enumeration member 986

PET_PTM_EVT_SUBSCRIBED enumeration member 986

PET_QUESTION_ALLOW_TO_SUSPEND_VM_WITH_BOOTCAMP_DISK 1116

PET_QUESTION_ALLOW_TO_SUSPEND_VM_WITH_BOOTCAMP_DISK macro 1116

PET_QUESTION_APPLIANCE_CORRUPTED_INSTALLATION 1116

PET_QUESTION_APPLIANCE_CORRUPTED_INSTALLATION macro 1116

PET_QUESTION_APPLY_WHEN_HYP_CANNOT_ALLOC_VMS_MEM 1116

PET_QUESTION_APPLY_WHEN_HYP_CANNOT_ALLOC_VMS_MEM macro 1116

PET_QUESTION_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE 1117

PET_QUESTION_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE macro 1117

PET_QUESTION_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE 1117

PET_QUESTION_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE macro 1117

PET_QUESTION_BOOTCAMP_HELPER_INIT_FAILURE 1117

PET_QUESTION_BOOTCAMP_HELPER_INIT_FAILURE macro 1117

PET_QUESTION_BOOTCAMP_HELPER_OS_UNSUPPORTED 1118

PET_QUESTION_BOOTCAMP_HELPER_OS_UNSUPPORTED macro 1118

PET_QUESTION_CANCEL_CLONE_OPERATION 1118

PET_QUESTION_CANCEL_CLONE_OPERATION macro 1118

PET_QUESTION_CANCEL_CLONE_TO_TEMPLATE_OPERATION 1118

PET_QUESTION_CANCEL_CLONE_TO_TEMPLATE_OPERATION macro 1118

PET_QUESTION_CANCEL_CONVERT_3RD_VM_OPERATION 1118

PET_QUESTION_CANCEL_CONVERT_3RD_VM_OPERATION macro 1118

PET_QUESTION_CANCEL_DEPLOY_OPERATION 1119

PET_QUESTION_CANCEL_DEPLOY_OPERATION macro 1119

PET_QUESTION_CANCEL_IMPORT_BOOTCAMP_OPERATION

1119
PET_QUESTION_CANCEL_IMPORT_BOOTCAMP_OPERATION
macro 1119
PET_QUESTION_CHANGE_VM_MIGRATION_TYPE 1119
PET_QUESTION_CHANGE_VM_MIGRATION_TYPE macro 1119
PET_QUESTION_COMMON_HDD_ERROR 1119
PET_QUESTION_COMMON_HDD_ERROR macro 1119
PET_QUESTION_COMPACT_VM_DISKS 1120
PET_QUESTION_COMPACT_VM_DISKS macro 1120
PET_QUESTION_CONTINUE_IF_HVT_DISABLED 1120
PET_QUESTION_CONTINUE_IF_HVT_DISABLED macro 1120
PET_QUESTION_CREATE_NEW_MAC_ADDRESS 1120
PET_QUESTION_CREATE_NEW_MAC_ADDRESS macro 1120
PET_QUESTION_CREATE_OS2_GUEST_WITHOUT_FDD_IMAGE
1120
PET_QUESTION_CREATE_OS2_GUEST_WITHOUT_FDD_IMAGE
macro 1120
PET_QUESTION_CREATE_VM_FROM_LION_RECOVERY_PART
1121
PET_QUESTION_CREATE_VM_FROM_LION_RECOVERY_PART
macro 1121
PET_QUESTION_DELAYED_COMPACT_VM_DISKS 1121
PET_QUESTION_DELAYED_COMPACT_VM_DISKS macro 1121
PET_QUESTION_DELETE_FILES_OUT_OF_VM_DIR 1121
PET_QUESTION_DELETE_FILES_OUT_OF_VM_DIR macro 1121
PET_QUESTION_DELETE_PARALLELS_FILES_IN_VM_DIR 1121
PET_QUESTION_DELETE_PARALLELS_FILES_IN_VM_DIR macro 1121
PET_QUESTION_DELETE_VM_WITH_CORRUPTED_CONFIG
1122
PET_QUESTION_DELETE_VM_WITH_CORRUPTED_CONFIG
macro 1122
PET_QUESTION_DO_YOU_WANT_TO_OVERWRITE_FILE 1122
PET_QUESTION_DO_YOU_WANT_TO_OVERWRITE_FILE macro 1122
PET_QUESTION_FORCE_COMPACT_VM_DISKS 1122
PET_QUESTION_FORCE_COMPACT_VM_DISKS macro 1122
PET_QUESTION_FREE_SIZE_FOR_COMPRESSED_DISK 1122
PET_QUESTION_FREE_SIZE_FOR_COMPRESSED_DISK macro 1122
PET_QUESTION_MAC_PHYSICAL_MEMORY_MAP_BUG 1123
PET_QUESTION_MAC_PHYSICAL_MEMORY_MAP_BUG macro 1123
PET_QUESTION_OLD_CONFIG_CONVERTION 1123
PET_QUESTION_OLD_CONFIG_CONVERTION macro 1123
PET_QUESTION_ON_QUERY_END_SESSION 1123
PET_QUESTION_ON_QUERY_END_SESSION macro 1123
PET_QUESTION_ON_QUERY_END_SESSION_RESTRICTED 1123
PET_QUESTION_ON_QUERY_END_SESSION_RESTRICTED macro 1123
PET_QUESTION_REACH_OVERCOMMIT_STATE 1124
PET_QUESTION_REACH_OVERCOMMIT_STATE macro 1124
PET_QUESTION_REACH_OVERCOMMIT_STATE_ON_SETMEM
1124
PET_QUESTION_REACH_OVERCOMMIT_STATE_ON_SETMEM
macro 1124
PET_QUESTION_REBOOT_HOST_ON_PCI_DRIVER_INSTALL_OR_REVERT
1124
PET_QUESTION_REBOOT_HOST_ON_PCI_DRIVER_INSTALL_OR_REVERT
macro 1124
PET_QUESTION_REGISTER_USED_VM 1124
PET_QUESTION_REGISTER_USED_VM macro 1124
PET_QUESTION_REGISTER_VM_TEMPLATE 1125
PET_QUESTION_REGISTER_VM_TEMPLATE macro 1125
PET_QUESTION_RESTART_VM_GUEST_TO_COMPACT 1125
PET_QUESTION_RESTART_VM_GUEST_TO_COMPACT macro 1125
PET_QUESTION_RESTORE_VM_CONFIG_FROM_BACKUP 1125
PET_QUESTION_RESTORE_VM_CONFIG_FROM_BACKUP

macro 1125

PET_QUESTION_SAMPLE_1 1125

PET_QUESTION_SAMPLE_1 macro 1125

PET_QUESTION_SAMPLE_2 1126

PET_QUESTION_SAMPLE_2 macro 1126

PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE 1126

PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE macro 1126

PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE_CPU 1126

PET_QUESTION_SNAPSHOT_STATE_INCOMPATIBLE_CPU macro 1126

PET_QUESTION_STOP_VM_TO_COMPACT 1126

PET_QUESTION_STOP_VM_TO_COMPACT macro 1126

PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE 1126

PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE macro 1126

PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE_CPU 1127

PET_QUESTION_SUSPEND_STATE_INCOMPATIBLE_CPU macro 1127

PET_QUESTION_SWITCH_OFF_AUTO_COMPRESS 1127

PET_QUESTION_SWITCH_OFF_AUTO_COMPRESS macro 1127

PET_QUESTION_TXT_ERROR 1127

PET_QUESTION_TXT_ERROR macro 1127

PET_QUESTION_UNDO_DISKS_MODE 1127

PET_QUESTION_UNDO_DISKS_MODE macro 1127

PET_QUESTION_VM_COPY_OR_MOVE 1128

PET_QUESTION_VM_COPY_OR_MOVE macro 1128

PET_QUESTION_VM_REBOOT_REQUIRED_BY_PRL_TOOLS 1128

PET_QUESTION_VM_REBOOT_REQUIRED_BY_PRL_TOOLS macro 1128

PET_QUESTION_VM_ROOT_DIRECTORY_NOT_EXISTS 1128

PET_QUESTION_VM_ROOT_DIRECTORY_NOT_EXISTS macro 1128

PET_VIDEO_CAPTURE_GEOMETRY enumeration member 986

PET_VIDEO_DATA_CHANNEL_CTL enumeration member 1016

PET_VIDEO_RECEIVER_CLOSED enumeration member 986

PET_VIDEO_RECEIVER_CONNECTION_STATE enumeration member 986

PET_VIDEO_RECEIVER_FORMAT enumeration member 986

PET_VIDEO_RECEIVER_FRAME enumeration member 986

PET_VIDEO_STREAM_ACK_FRAME enumeration member 1016

PET_VIDEO_STREAM_CLOSE enumeration member 1016

PET_VIDEO_STREAM_CLOSED enumeration member 1016

PET_VIDEO_STREAM_DATA enumeration member 1016

PET_VIDEO_STREAM_FORMAT enumeration member 1016

PET_VIDEO_STREAM_OPEN enumeration member 1016

PET_VIDEO_STREAM_START enumeration member 1016

PET_VIDEO_STREAM_STOP enumeration member 1016

PET_VM_DEFAULT_SOUND_RECONNECTED enumeration member 986

PET_VM_EVT_MOUNT enumeration member 986

PET_VM_EVT_REQUEST_FILE_OPEN enumeration member 986

PET_VM_EVT_TIS_BACKUP_WRITE enumeration member 986

PET_VM_EVT_UNMOUNT enumeration member 986

PET_VM_EVT_UNMOUNT_DVD_MEDIA enumeration member 986

PET_VM_EVT_UNMOUNT_USB_MEDIA enumeration member 986

PET_VM_EVT_VM_CONFIG_CHANGED enumeration member 986

PET_VM_INF_END_BUNCH_COPYING enumeration member 986

PET_VM_INF_END_FILE_COPYING enumeration member 986

PET_VM_INF_START_BUNCH_COPYING enumeration member 986

PET_VM_INF_START_FILE_COPYING enumeration member 986

PET_VM_INF_UNINITIALIZED_EVENT_CODE enumeration member 986

PET_VM_READY_TO_BOOT enumeration member 986

PET_VM_READY_TO_PRINT enumeration member 986

PEVF_WAIT_FOR_APPLY enumeration member 927

PEVT_NATIVE_VM enumeration member 991

PEVT_VM_UNKNOWN enumeration member 991

PEVT_VMWARE enumeration member 991

PFD_ALL enumeration member 929

PFD_BINARY enumeration member 1015

PFD_BOOLEAN enumeration member 1015
PFD_CDATA enumeration member 1015
PFD_ENTITY enumeration member 1015
PFD_ENUMERATION enumeration member 1015
PFD_INCOMING enumeration member 993
PFD_INT32 enumeration member 1015
PFD_INT64 enumeration member 1015
PFD_OUTGOING enumeration member 993
PFD_STDERR enumeration member 929
PFD_STDIN enumeration member 929
PFD_STDOUT enumeration member 929
PFD_STRING enumeration member 1015
PFD_UINT32 enumeration member 1015
PFD_UINT64 enumeration member 1015
PFD_UNKNOWN enumeration member 1015
PFP_ACCEPT enumeration member 993
PFP_DENY enumeration member 993
PFS_UNIX_LIKE_FS enumeration member 992
PFS_WINDOWS_LIKE_FS enumeration member 992
PFSM_AUTOSTART_VM_AS_OWNER enumeration member 991
PFSM_CPU_HOTPLUG_SUPPORT enumeration member 991
PFSM_DEFAULT_PLAINDISK_ALLOWED enumeration member 991
PFSM_DEFAULT_SATA_ALLOWED enumeration member 991
PFSM_DESKTOP_CONTROL_SUPPORT enumeration member 991
PFSM_DISK_IO_LIMITS enumeration member 991
PFSM_FINE_CPU_LIMITS enumeration member 991
PFSM_IPV6_SUPPORT enumeration member 991
PFSM_NIC_CHANGE_ALLOWED enumeration member 991
PFSM_NO_SHARED_NETWORKING enumeration member 991
PFSM_PSBM5 enumeration member 991
PFSM_RAM_HOTPLUG_SUPPORT enumeration member 991
PFSM_ROUTED_NETWORKING enumeration member 991
PFSM_SATA_HOTPLUG_SUPPORT enumeration member 991
PFSM_UNKNOWN_FEATURE enumeration member 991
PFSM_USB_PRINTER_SUPPORT enumeration member 991
PFSM_VM_CONFIG_MERGE_SUPPORT enumeration member 991
PGD_PCI_DISPLAY enumeration member 994
PGD_PCI_NETWORK enumeration member 994
PGD_PCI_OTHER enumeration member 994
PGD_PCI_SOUND enumeration member 994
PGD_UNKNOWN enumeration member 996
PGD_WINDOWS_VISTA enumeration member 996
PGD_WINDOWS_VISTA_32 enumeration member 996
PGD_WINDOWS_VISTA_64 enumeration member 996
PGD_WINDOWS_XP enumeration member 996
PGD_WINDOWS_XP_32 enumeration member 996
PGD_WINDOWS_XP_64 enumeration member 996
PGS_CONNECTED_TO_HOST enumeration member 993
PGS_CONNECTED_TO_VM enumeration member 993
PGS_CONNECTING_TO_VM enumeration member 993
PGS_NON_CONTROLLED_USB enumeration member 993
PGS_OS_EXPERIMENTAL enumeration member 996
PGS_OS_SUPPORTED enumeration member 996
PGS_RESERVED enumeration member 993
PGST_CAPABILITY_HIBERNATE enumeration member 995
PGST_CAPABILITY_REBOOT enumeration member 995
PGST_CAPABILITY_SHUTDOWN enumeration member 995
PGST_CAPABILITY_SUSPEND enumeration member 995
PGST_FULL_SIZE_SCREENSHOTS enumeration member 994
PGST_WITHOUT_SCREENSHOTS enumeration member 994
PGVC_FILL_AUTOGENERATED enumeration member 994
PGVC_SEARCH_BY_NAME enumeration member 994
PGVC_SEARCH_BY_UUID enumeration member 994
PGVLF_FILL_AUTOGENERATED enumeration member 995
PGVLF_GET_ONLY_CT enumeration member 995
PGVLF_GET_ONLY_IDENTITY_INFO enumeration member 995
PGVLF_GET_ONLY_VM enumeration member 995
PGVLF_GET_ONLY_VM_TEMPLATES enumeration member 995
PGVLF_GET_STATE_INFO enumeration member 995
PHD_EXPANDING_HARD_DISK enumeration member 999
PHD_PLAIN_HARD_DISK enumeration member 999
PHI_REAL_NET_ADAPTER enumeration member 999
PHI_VIRTUAL_NET_ADAPTER enumeration member 999
PHO_LIN enumeration member 999

PHO_MAC enumeration member 999
PHO_UNKNOWN enumeration member 999
PHO_WIN enumeration member 999
PHT_ACCESS_RIGHTS 9
PHT_ACCESS_RIGHTS enumeration member 996
PHT_APPLIANCE_CONFIG enumeration member 996
PHT_BACKUP 13
PHT_BACKUP_RESULT 13
PHT_BACKUP_RESULT enumeration member 996
PHT_BOOT_DEVICE 14
PHT_BOOT_DEVICE enumeration member 996
PHT_CPU_FEATURES 22
PHT_CPU_FEATURES enumeration member 996
PHT_CPU_POOL 24
PHT_CPU_POOL enumeration member 996
PHT_CT_TEMPLATE 26
PHT_CT_TEMPLATE enumeration member 996
PHT_CVSRC 32
PHT_CVSRC enumeration member 996
PHT_DESKTOP_CONTROL 32
PHT_DESKTOP_CONTROL enumeration member 996
PHT_DISP_CONFIG 32
PHT_DISP_CONFIG enumeration member 996
PHT_DISP_NET_ADAPTER enumeration member 996
PHT_ERROR enumeration member 996
PHT_EVENT 72
PHT_EVENT enumeration member 996
PHT_EVENT_PARAMETER 88
PHT_EVENT_PARAMETER enumeration member 996
PHT_FIREWALL_RULE 95
PHT_FIREWALL_RULE enumeration member 996
PHT_FOUND_VM_INFO 102
PHT_FOUND_VM_INFO enumeration member 996
PHT_GUEST_OSES_MATRIX 105
PHT_GUEST_OSES_MATRIX enumeration member 996
PHT_HANDLES_LIST 108
PHT_HANDLES_LIST enumeration member 996
PHT_HW_GENERIC_DEVICE 111
PHT_HW_GENERIC_DEVICE enumeration member 996
PHT_HW_GENERIC_PCI_DEVICE 115
PHT_HW_GENERIC_PCI_DEVICE enumeration member 996
PHT_HW_HARD_DISK 117
PHT_HW_HARD_DISK enumeration member 996
PHT_HW_HARD_DISK_PARTITION 121
PHT_HW_HARD_DISK_PARTITION enumeration member 996
PHT_HW_NET_ADAPTER 126
PHT_HW_NET_ADAPTER enumeration member 996
PHT_IPPRIV_NET 133
PHT_IPPRIV_NET enumeration member 996
PHT_ISCSI_LUN enumeration member 996
PHT_JOB 138
PHT_JOB enumeration member 996
PHT_LAST enumeration member 996
PHT_LICENSE 146
PHT_LICENSE enumeration member 996
PHT_LOGIN_RESPONSE 149
PHT_LOGIN_RESPONSE enumeration member 996
PHT_NET_SERVICE_STATUS 153
PHT_NET_SERVICE_STATUS enumeration member 996
PHT_NETWORK_CLASS 154
PHT_NETWORK_CLASS enumeration member 996
PHT_NETWORK_RATE 157
PHT_NETWORK_RATE enumeration member 996
PHT_NETWORK_SHAPING 160
PHT_NETWORK_SHAPING enumeration member 996
PHT_NETWORK_SHAPING_BANDWIDTH 166
PHT_NETWORK_SHAPING_BANDWIDTH enumeration member 996
PHT_NETWORK_SHAPING_CONFIG 169
PHT_NETWORK_SHAPING_CONFIG enumeration member 996
PHT_OFFLINE_SERVICE 173
PHT_OFFLINE_SERVICE enumeration member 996
PHT_OPAQUE_TYPE_LIST 177
PHT_OPAQUE_TYPE_LIST enumeration member 996
PHT_PLUGIN_INFO 180
PHT_PLUGIN_INFO enumeration member 996
PHT_PORT_FORWARDING 185
PHT_PORT_FORWARDING enumeration member 996

PHT_PROBLEM_REPORT 192
PHT_PROBLEM_REPORT enumeration member 996
PHT_REMOTE_FILESYSTEM_ENTRY 202
PHT_REMOTE_FILESYSTEM_ENTRY enumeration member 996
PHT_REMOTE_FILESYSTEM_INFO 206
PHT_REMOTE_FILESYSTEM_INFO enumeration member 996
PHT_REMOTEDEV_CMD enumeration member 996
PHT_RESULT 210
PHT_RESULT enumeration member 996
PHT_RUNNING_TASK 217
PHT_RUNNING_TASK enumeration member 996
PHT_SCREEN_RESOLUTION 219
PHT_SCREEN_RESOLUTION enumeration member 996
PHT_SERVER 219
PHT_SERVER enumeration member 996
PHT_SERVER_CONFIG 314
PHT_SERVER_CONFIG enumeration member 996
PHT_SERVER_INFO 344
PHT_SERVER_INFO enumeration member 996
PHT_SESSION_INFO 350
PHT_SESSION_INFO enumeration member 996
PHT_SHARE 350
PHT_SHARE enumeration member 996
PHT_STRINGS_LIST 357
PHT_STRINGS_LIST enumeration member 996
PHT_SYSTEM_STATISTICS 360
PHT_SYSTEM_STATISTICS enumeration member 996
PHT_SYSTEM_STATISTICS_CPU 365
PHT_SYSTEM_STATISTICS_CPU enumeration member 996
PHT_SYSTEM_STATISTICS_DISK 367
PHT_SYSTEM_STATISTICS_DISK enumeration member 996
PHT_SYSTEM_STATISTICS_DISK_PARTITION 367
PHT_SYSTEM_STATISTICS_DISK_PARTITION enumeration member 996
PHT_SYSTEM_STATISTICS_IFACE 368
PHT_SYSTEM_STATISTICS_IFACE enumeration member 996
PHT_SYSTEM_STATISTICS_PROCESS 368
PHT_SYSTEM_STATISTICS_PROCESS enumeration member 996
PHT_SYSTEM_STATISTICS_USER_SESSION 368
PHT_SYSTEM_STATISTICS_USER_SESSION enumeration member 996
PHT_SYSTEM_STATISTICS_VM_DATA 368
PHT_SYSTEM_STATISTICS_VM_DATA enumeration member 996
PHT_TIS_EMITTER enumeration member 996
PHT_TIS_RECORD enumeration member 996
PHT_TOOL enumeration member 996
PHT_UIEMU_INPUT 368
PHT_USB_IDENTITY 368
PHT_USB_IDENTITY enumeration member 996
PHT_USER_INFO 371
PHT_USER_INFO enumeration member 996
PHT_USER_PROFILE 374
PHT_USER_PROFILE enumeration member 996
PHT_VCMMD_CONFIG 379
PHT_VCMMD_CONFIG enumeration member 996
PHT_VIDEO_RECEIVER 381
PHT_VIDEO_RECEIVER enumeration member 996
PHT_VIRTUAL_DEV_DISPLAY enumeration member 996
PHT_VIRTUAL_DEV_FLOPPY 381
PHT_VIRTUAL_DEV_FLOPPY enumeration member 996
PHT_VIRTUAL_DEV_GENERIC_PCI enumeration member 996
PHT_VIRTUAL_DEV_GENERIC_SCSI enumeration member 996
PHT_VIRTUAL_DEV_HARD_DISK 381
PHT_VIRTUAL_DEV_HARD_DISK enumeration member 996
PHT_VIRTUAL_DEV_HD_PARTITION 397
PHT_VIRTUAL_DEV_HD_PARTITION enumeration member 996
PHT_VIRTUAL_DEV_KEYBOARD enumeration member 996
PHT_VIRTUAL_DEV_MOUSE enumeration member 996
PHT_VIRTUAL_DEV_NET_ADAPTER 399
PHT_VIRTUAL_DEV_NET_ADAPTER enumeration member 996
PHT_VIRTUAL_DEV_OPTICAL_DISK 425
PHT_VIRTUAL_DEV_OPTICAL_DISK enumeration member 996
PHT_VIRTUAL_DEV_PARALLEL_PORT 425
PHT_VIRTUAL_DEV_PARALLEL_PORT enumeration member 996

PHT_VIRTUAL_DEV_SERIAL_PORT 425
PHT_VIRTUAL_DEV_SERIAL_PORT enumeration member 996
PHT_VIRTUAL_DEV_SOUND 427
PHT_VIRTUAL_DEV_SOUND enumeration member 996
PHT_VIRTUAL_DEV_USB_DEVICE 430
PHT_VIRTUAL_DEV_USB_DEVICE enumeration member 996
PHT_VIRTUAL_DISK 432
PHT_VIRTUAL_DISK enumeration member 996
PHT_VIRTUAL_DISK_ENCRYPTION 440
PHT_VIRTUAL_DISK_ENCRYPTION enumeration member 996
PHT_VIRTUAL_DISK_MAP enumeration member 996
PHT_VIRTUAL_DISK_OPEN_POLICY enumeration member 996
PHT_VIRTUAL_MACHINE 442
PHT_VIRTUAL_MACHINE enumeration member 996
PHT_VIRTUAL_NETWORK 507
PHT_VIRTUAL_NETWORK enumeration member 996
PHT_VM_CONFIGURATION 537
PHT_VM_CONFIGURATION enumeration member 996
PHT_VM_GUEST_SESSION 716
PHT_VM_GUEST_SESSION enumeration member 996
PHT_VM_INFO 719
PHT_VM_INFO enumeration member 996
PHT_VM_TOOLS_INFO 724
PHT_VM_TOOLS_INFO enumeration member 996
PHT_VM_VIRTUAL_DEVICES_INFO 726
PHT_VM_VIRTUAL_DEVICES_INFO enumeration member 996
PHY_WIFI_REAL_NET_ADAPTER enumeration member 999
PIAF_CANCEL enumeration member 1000
PIAF_FORCE enumeration member 1000
PIAF_STOP enumeration member 1000
PIE_DISPATCHER enumeration member 986
PIE_IO_SERVICE enumeration member 986
PIE_UNKNOWN enumeration member 986
PIE_VIRTUAL_MACHINE enumeration member 986
PIE_WEB_SERVICE enumeration member 986
PIF_BMP enumeration member 1000
PIF_JPG enumeration member 1000
PIF_PNG enumeration member 1000
PIF_RAW enumeration member 1000
PIM_ALL_DISKS enumeration member 998
PIM_COMPACT_DELAYED enumeration member 998
PIM_IDE_0_0 enumeration member 998
PIM_IDE_0_1 enumeration member 998
PIM_IDE_1_0 enumeration member 998
PIM_IDE_1_1 enumeration member 998
PIM_IDE_MASK_OFFSET enumeration member 998
PIM_SATA_0_0 enumeration member 998
PIM_SATA_0_1 enumeration member 998
PIM_SATA_0_2 enumeration member 998
PIM_SATA_0_3 enumeration member 998
PIM_SATA_0_4 enumeration member 998
PIM_SATA_0_5 enumeration member 998
PIM_SATA_MASK_OFFSET enumeration member 998
PIM_SCSI_0_0 enumeration member 998
PIM_SCSI_1_0 enumeration member 998
PIM_SCSI_10_0 enumeration member 998
PIM_SCSI_11_0 enumeration member 998
PIM_SCSI_12_0 enumeration member 998
PIM_SCSI_13_0 enumeration member 998
PIM_SCSI_14_0 enumeration member 998
PIM_SCSI_15_0 enumeration member 998
PIM_SCSI_2_0 enumeration member 998
PIM_SCSI_3_0 enumeration member 998
PIM_SCSI_4_0 enumeration member 998
PIM_SCSI_5_0 enumeration member 998
PIM_SCSI_6_0 enumeration member 998
PIM_SCSI_7_0 enumeration member 998
PIM_SCSI_8_0 enumeration member 998
PIM_SCSI_9_0 enumeration member 998
PIM_SCSI_MASK_OFFSET enumeration member 998
PISCSI_STORAGE_CREATE enumeration member 1001
PISCSI_STORAGE_EXT3 enumeration member 1001
PISCSI_STORAGE_EXT4 enumeration member 1001
PISCSI_STORAGE_MOUNT enumeration member 1001
PISCSI_STORAGE_MOUNT_RDONLY enumeration member 1001
PISCSI_STORAGE_NTFS enumeration member 1001
PISCSI_STORAGE_REMOUNT enumeration member 1001
PISCSI_STORAGE_REMOVE enumeration member 1001

PIT_DOCK_ICON_LIVE_SCREEN_SHOT enumeration member 985

PIT_DOCK_ICON_START_MENU enumeration member 985

PIT_DOCK_ICON_SYSTEM enumeration member 985

PJOC_API_GET_USER_DATA enumeration member 1002

PJOC_API_GET_USER_KEYS enumeration member 1002

PJOC_API_PUT_ICON enumeration member 1002

PJOC_API_PUT_USER_DATA enumeration member 1002

PJOC_API_REMOVE_USER_DATA enumeration member 1002

PJOC_API_SEND_PACKED_PROBLEM_REPORT enumeration member 1002

PJOC_API_SEND_PROBLEM_REPORT enumeration member 1002

PJOC_API_SUBSCRIBE_ON_REGISTRATION enumeration member 1002

PJOC_API_VALIDATE_IN_CACHE enumeration member 1002

PJOC_CT_REINSTALL enumeration member 1002

PJOC_DESKTOP_CONTROL_CONNECT enumeration member 1002

PJOC_JOB_CANCEL enumeration member 1002

PJOC_PTM_RPC enumeration member 1002

PJOC_REPORT_ASSEMBLY enumeration member 1002

PJOC_SRV_ADD_IPPRIVATE_NETWORK enumeration member 1002

PJOC_SRV_ADD_NET_ADAPTER enumeration member 1002

PJOC_SRV_ADD_VIRTUAL_NETWORK enumeration member 1002

PJOC_SRV_AFTER_HOST_RESUME enumeration member 1002

PJOC_SRV_ATTACH_TO_LOST_TASK enumeration member 1002

PJOC_SRV_CHECK_ALIVE enumeration member 1002

PJOC_SRV_COMMON_PREFS_BEGIN_EDIT enumeration member 1002

PJOC_SRV_COMMON_PREFS_COMMIT enumeration member 1002

PJOC_SRV_CONFIGURE_GENERIC_PCI enumeration member 1002

PJOC_SRV_COPY_CT_TEMPLATE enumeration member 1002

PJOC_SRV_CPU_POOLS_JOIN enumeration member 1002

PJOC_SRV_CPU_POOLS_LEAVE enumeration member 1002

PJOC_SRV_CPU_POOLS_LIST_POOLS enumeration member 1002

PJOC_SRV_CPU_POOLS_MOVE enumeration member 1002

PJOC_SRV_CPU_POOLS_RECALCULATE enumeration member 1002

PJOC_SRV_CREATE_UNATTENDED_CD enumeration member 1002

PJOC_SRV_CREATE_VM_BACKUP enumeration member 1002

PJOC_SRV_DELETE_NET_ADAPTER enumeration member 1002

PJOC_SRV_DELETE_OFFLINE_SERVICE enumeration member 1002

PJOC_SRV_DELETE_VIRTUAL_NETWORK enumeration member 1002

PJOC_SRV_EXTEND_ISCSI_STORAGE enumeration member 1002

PJOC_SRV_FS_CAN_CREATE_FILE enumeration member 1002

PJOC_SRV_FS_CREATE_DIR enumeration member 1002

PJOC_SRV_FS_GENERATE_ENTRY_NAME enumeration member 1002

PJOC_SRV_FS_GET_DIR_ENTRIES enumeration member 1002

PJOC_SRV_FS_GET_DISK_LIST enumeration member 1002

PJOC_SRV_FS_REMOVE_ENTRY enumeration member 1002

PJOC_SRV_FS_RENAME_ENTRY enumeration member 1002

PJOC_SRV_GENERATE_CSR enumeration member 1002

PJOC_SRV_GET_ALL_HOST_USERS enumeration member 1002

PJOC_SRV_GET_BACKUP_TREE enumeration member 1002

PJOC_SRV_GET_COMMON_PREFS enumeration member 1002

PJOC_SRV_GET_CT_TEMPLATE_LIST enumeration member 1002

PJOC_SRV_GET_DEFAULT_VM_CONFIG enumeration member 1002

PJOC_SRV_GET_DISK_FREE_SPACE enumeration member 1002

PJOC_SRV_GET_IPPRIVATE_NETWORKS_LIST enumeration member 1002

PJOC_SRV_GET_LICENSE_INFO enumeration member 1002

PJOC_SRV_GET_NET_SERVICE_STATUS enumeration member 1002

PJOC_SRV_GET_NETWORK_CLASSES_LIST enumeration member 1002

PJOC_SRV_GET_NETWORK_SHAPING_CONFIG enumeration member 1002

PJOC_SRV_GET_OFFLINE_SERVICES_LIST enumeration member 1002

PJOC_SRV_GET_PACKED_PROBLEM_REPORT enumeration member 1002

PJOC_SRV_GET_PERFSTATS enumeration member 1002

PJOC_SRV_GET_PLUGINS_LIST enumeration member 1002

PJOC_SRV_GET_PROBLEM_REPORT enumeration member 1002

PJOC_SRV_GET_SRV_CONFIG enumeration member 1002

PJOC_SRV_GET_STATISTICS enumeration member 1002

PJOC_SRV_GET_USER_INFO enumeration member 1002

PJOC_SRV_GET_USER_INFO_LIST enumeration member 1002

PJOC_SRV_GET_USER_PROFILE enumeration member 1002

PJOC_SRV_GET_VCMMD_CONFIG enumeration member 1002

PJOC_SRV_GET_VIRTUAL_NETWORK_LIST enumeration member 1002

PJOC_SRV_GET_VM_CONFIG enumeration member 1002

PJOC_SRV_GET_VM_LIST enumeration member 1002

PJOC_SRV_INSTALL_APPLIANCE enumeration member 1002

PJOC_SRV_LOGIN enumeration member 1002

PJOC_SRV_LOGIN_LOCAL enumeration member 1002

PJOC_SRV_LOGOFF enumeration member 1002

PJOC_SRV_LOOKUP_PARALLELS_SERVERS enumeration member 1002

PJOC_SRV_NET_SERVICE_RESTART enumeration member 1002

PJOC_SRV_NET_SERVICE_RESTORE_DEFAULTS enumeration member 1002

PJOC_SRV_NET_SERVICE_START enumeration member 1002

PJOC_SRV_NET_SERVICE_STOP enumeration member 1002

PJOC_SRV_PREPARE_FOR_HIBERNATE enumeration member 1002

PJOC_SRV_PROXY_GET_REGISTERED_HOSTS enumeration member 1002

PJOC_SRV_REFRESH_PLUGINS enumeration member 1002

PJOC_SRV_REGISTER_3RD_PARTY_VM enumeration member 1002

PJOC_SRV_REGISTER_ISCSI_STORAGE enumeration member 1002

PJOC_SRV_REGISTER_VM enumeration member 1002

PJOC_SRV_REGISTER_VM_EX enumeration member 1002

PJOC_SRV_REMOVE_CT_TEMPLATE enumeration member 1002

PJOC_SRV_REMOVE_IPPRIVATE_NETWORK enumeration member 1002

PJOC_SRV_REMOVE_VM_BACKUP enumeration member 1002

PJOC_SRV_RESTART_NETWORK_SHAPING enumeration member 1002

PJOC_SRV_RESTORE_VM_BACKUP enumeration member 1002

PJOC_SRV_SEND_ANSWER enumeration member 1002

PJOC_SRV_SEND_CLIENT_STATISTICS enumeration member 1002

PJOC_SRV_SEND_PROBLEM_REPORT enumeration member 1002

PJOC_SRV_SET_NON_INTERACTIVE_SESSION enumeration member 1002

PJOC_SRV_SET_SESSION_CONFIRMATION_MODE enumeration member 1002

PJOC_SRV_SHUTDOWN enumeration member 1002

PJOC_SRV_SIGN_CERTIFICATION_REQUEST enumeration member 1002

PJOC_SRV_START_CLUSTER_SERVICE enumeration member 1002

PJOC_SRV_START_SEARCH_VMS enumeration member 1002

PJOC_SRV_STOP_CLUSTER_SERVICE enumeration member 1002

PJOC_SRV_STORE_VALUE_BY_KEY enumeration member 1002

PJOC_SRV_SUBSCRIBE_PERFSTATS enumeration member 1002

PJOC_SRV_SUBSCRIBE_TO_HOST_STATISTICS enumeration member 1002

PJOC_SRV_UNREGISTER_ISCSI_STORAGE enumeration member 1002

PJOC_SRV_UNSUBSCRIBE_FROM_HOST_STATISTICS enumeration member 1002

PJOC_SRV_UNSUBSCRIBE_PERFSTATS enumeration member 1002

PJOC_SRV_UPDATE_IPPRIVATE_NETWORK enumeration member 1002

PJOC_SRV_UPDATE_LICENSE enumeration member 1002

PJOC_SRV_UPDATE_NET_ADAPTER enumeration member 1002

PJOC_SRV_UPDATE_NETWORK_CLASSES_CONFIG enumeration member 1002

PJOC_SRV_UPDATE_NETWORK_SHAPING_CONFIG enumeration member 1002

PJOC_SRV_UPDATE_OFFLINE_SERVICE enumeration member 1002

PJOC_SRV_UPDATE_SESSION_INFO enumeration member 1002

PJOC_SRV_UPDATE_USB_ASSOC_LIST enumeration member 1002

PJOC_SRV_UPDATE_VIRTUAL_NETWORK enumeration member 1002

PJOC_SRV_USER_PROFILE_BEGIN_EDIT enumeration member 1002

PJOC_SRV_USER_PROFILE_COMMIT enumeration member 1002

PJOC_UNKNOWN enumeration member 1002

PJOC_VM_AUTH_WITH_GUEST_SECURITY_DB enumeration member 1002

PJOC_VM_BEGIN_EDIT enumeration member 1002

PJOC_VM_CANCEL_COMPACT enumeration member 1002

PJOC_VM_CANCEL_COMPRESSOR enumeration member 1002

PJOC_VM_CHANGE_SID enumeration member 1002

PJOC_VM_CLONE enumeration member 1002

PJOC_VM_CMD_INTERNAL enumeration member 1002

PJOC_VM_COMMIT enumeration member 1002

PJOC_VM_COMMIT_ENCRYPTION enumeration member 1002

PJOC_VM_COMPACT enumeration member 1002

PJOC_VM_CONNECT_TO_VM enumeration member 1002

PJOC_VM_CONVERT_DISKS enumeration member 1002

PJOC_VM_CREATE_SNAPSHOT enumeration member 1002

PJOC_VM_CREATE_UNATTENDED_FLOPPY enumeration member 1002

PJOC_VM_DELETE enumeration member 1002

PJOC_VM_DELETE_SNAPSHOT enumeration member 1002

PJOC_VM_DEV_CONNECT enumeration member 1002

PJOC_VM_DEV_COPY_IMAGE enumeration member 1002

PJOC_VM_DEV_CREATE_IMAGE enumeration member 1002

PJOC_VM_DEV_DISCONNECT enumeration member 1002

PJOC_VM_DEV_DISPLAY_CAPTURE_SCREEN enumeration member 1002

PJOC_VM_DROP_SUSPENDED_STATE enumeration member 1002

PJOC_VM_GENERATE_VM_DEV_FILENAME enumeration member 1002

PJOC_VM_GET_PACKED_PROBLEM_REPORT enumeration member 1002

PJOC_VM_GET_PERFSTATS enumeration member 1002

PJOC_VM_GET_PROBLEM_REPORT enumeration member 1002

PJOC_VM_GET_SNAPSHOTS_TREE enumeration member 1002

PJOC_VM_GET_STATE enumeration member 1002

PJOC_VM_GET_STATISTICS enumeration member 1002

PJOC_VM_GET_SUSPENDED_SCREEN enumeration member 1002

PJOC_VM_GET_TOOLS_STATE enumeration member 1002

PJOC_VM_GET_VIRTUAL_DEVICES_INFO enumeration member 1002

PJOC_VM_GUEST_GET_NETWORK_SETTINGS enumeration member 1002

PJOC_VM_GUEST_LOGOUT enumeration member 1002

PJOC_VM_GUEST_RUN_PROGRAM enumeration member 1002

PJOC_VM_GUEST_SET_USER_PASSWD enumeration member 1002

PJOC_VM_INITIATE_DEV_STATE_NOTIFICATIONS enumeration member 1002

PJOC_VM_INSTALL_TOOLS enumeration member 1002

PJOC_VM_INSTALL_UTILITY enumeration member 1002

PJOC_VM_LOCK enumeration member 1002

PJOC_VM_LOGIN_IN_GUEST enumeration member 1002

PJOC_VM_MIGRATE enumeration member 1002

PJOC_VM_MIGRATE_CANCEL enumeration member 1002

PJOC_VM_MOUNT enumeration member 1002

PJOC_VM_MOVE enumeration member 1002

PJOC_VM_PAUSE enumeration member 1002

PJOC_VM_REFRESH_CONFIG enumeration member 1002

PJOC_VM_REG enumeration member 1002

PJOC_VM_REMOVE_PROTECTION enumeration member 1002

PJOC_VM_RESET enumeration member 1002

PJOC_VM_RESET_UPTIME enumeration member 1002

PJOC_VM_RESIZE_DISK_IMAGE enumeration member 1002

PJOC_VM_RESTART enumeration member 1002

PJOC_VM_RESTORE enumeration member 1002

PJOC_VM_RESUME enumeration member 1002

PJOC_VM_RUN_COMPRESSOR enumeration member 1002

PJOC_VM_SEND_CLIPBOARD_REQUEST enumeration member 1002

PJOC_VM_SEND_DRAGDROP_COMMAND enumeration member 1002

PJOC_VM_SEND_PROBLEM_REPORT enumeration member 1002

PJOC_VM_SEND_SHUTDOWN_COMMAND enumeration member 1002

PJOC_VM_SET_PROTECTION enumeration member 1002

PJOC_VM_START enumeration member 1002

PJOC_VM_START_EX enumeration member 1002

PJOC_VM_START_VNC_SERVER enumeration member 1002

PJOC_VM_STOP enumeration member 1002

PJOC_VM_STOP_VNC_SERVER enumeration member 1002

PJOC_VM_STORE_VALUE_BY_KEY enumeration member 1002

PJOC_VM_SUBSCRIBE_PERFSTATS enumeration member 1002

PJOC_VM_SUBSCRIBE_TO_GUEST_STATISTICS enumeration member 1002

PJOC_VM_SUSPEND enumeration member 1002

PJOC_VM_SWITCH_TO_SNAPSHOT enumeration member 1002

PJOC_VM_UMOUNT enumeration member 1002

PJOC_VM_UNLOCK enumeration member 1002

PJOC_VM_UNREG enumeration member 1002

PJOC_VM_UNSUBSCRIBE_FROM_GUEST_STATISTICS enumeration member 1002

PJOC_VM_UNSUBSCRIBE_PERFSTATS enumeration member 1002

PJOC_VM_UPDATE_SECURITY enumeration member 1002

PJOC_VM_UPDATE_SNAPSHOT_DATA enumeration member 1002

PJOC_VM_UPDATE_TOOLS_SECTION enumeration member 1002

PJOC_VM_VALIDATE_CONFIG enumeration member 1002

PJS_FINISHED enumeration member 1005

PJS_RUNNING enumeration member 1005

PJS_UNKNOWN enumeration member 1005

PKE_PRESS enumeration member 1008

PKE_RELEASE enumeration member 1008

PLLF_LOGIN_WITH_DELAYED_CONNECTION enumeration member 1010

PLM_ROOT_ACCOUNT enumeration member 934

PLM_ROOT_ACOUNT enumeration member 934

PLM_START_ACCOUNT enumeration member 934

PLM_START_ACOUNT enumeration member 934

PLM_USER_ACCOUNT enumeration member 934

PLM_USER_ACOUNT enumeration member 934

PLRK_ALLOWED_GUEST_OS_VERSIONS enumeration member 1009

PLRK_RUNNING_VMS_LIMIT enumeration member 1009

PLRK_RUNNING_VMS_LIMIT_PER_USER enumeration member 1009

PLRK_UNKNOWN enumeration member 1009

PLRK_VM_CLONE enumeration member 1009

PLRK_VM_CONVERT_FROM_TEMPLATE enumeration member 1009

PLRK_VM_CONVERT_TO_TEMPLATE enumeration member 1009

PLRK_VM_CPU_LIMIT enumeration member 1009

PLRK_VM_CREATE enumeration member 1009

PLRK_VM_DOWNLOAD enumeration member 1009

PLRK_VM_IMPORT_3RD_PARTY enumeration member 1009

PLRK_VM_MEMORY_LIMIT enumeration member 1009

PLRK_VM_PAUSE enumeration member 1009

PLRK_VM_REGISTER enumeration member 1009

PLRK_VM_SAFEMODE_FEATURE enumeration member 1009

PLRK_VM_SHOW_FULLSCREEN enumeration member 1009

PLRK_VM_SMARTGUARD_FEATURE enumeration member 1009

PLRK_VM_SNAPSHOT_CREATE enumeration member 1009

PLRK_VM_SNAPSHOT_DELETE enumeration member 1009

PLRK_VM_SNAPSHOT_SWITCH enumeration member 1009

PLRK_VM_SNAPSHOT_TREE enumeration member 1009

PLRK_VM_SUSPEND enumeration member 1009

PLRK_VM_UNDODISK_FEATURE enumeration member 1009

PLRK_VM_VTD_AVAILABLE enumeration member 1009

PMAA_NO_ADVANCED_AUTH_NEEDED enumeration member 1012

PMAA_USE_SYSTEM_CREDENTIALS enumeration member 1012

PMB_LEFT_BUTTON enumeration member 1012

PMB_MIDDLE_BUTTON enumeration member 1012

PMB_NOBUTTON enumeration member 1012

PMB_RIGHT_BUTTON enumeration member 1012
PMB_XBUTTON1 enumeration member 1012
PMB_XBUTTON2 enumeration member 1012
PMB_XBUTTON3 enumeration member 1012
PMB_XBUTTON4 enumeration member 1012
PMB_XBUTTON5 enumeration member 1012
PMS_IDE_DEVICE enumeration member 1011
PMS_SATA_DEVICE enumeration member 1011
PMS_SCSI_DEVICE enumeration member 1011
PMS_UNKNOWN_DEVICE enumeration member 1011
PMS_VIRTIO_BLOCK_DEVICE enumeration member 1011
PMT_ANSWER enumeration member 1011
PMT_CRITICAL enumeration member 1011
PMT_INFORMATION enumeration member 1011
PMT_QUESTION enumeration member 1011
PMT_WARNING enumeration member 1011
PMVD_INFO enumeration member 1012
PMVD_READ_ONLY enumeration member 1012
PMVD_READ_WRITE enumeration member 1012
PNA_BRIDGE enumeration member 1013
PNA_BRIDGED_ETHERNET enumeration member 1013
PNA_BRIDGED_NETWORK enumeration member 1013
PNA_DIRECT_ASSIGN enumeration member 1013
PNA_HOST_ONLY enumeration member 1013
PNA_ROUTED enumeration member 1013
PNA_SHARED enumeration member 1013
PNSF_CT_SKIP_ARPDETECT enumeration member 933
PNSF_VM_START_WAIT enumeration member 933
PNT_E1000 enumeration member 932
PNT_E1000E enumeration member 932
PNT_RTL enumeration member 932
PNT_UNDEFINED enumeration member 932
PNT_VIRTIO enumeration member 932
PNWR_AUTO enumeration member 933
PNWR_GENERATED_MAC enumeration member 933
PNWR_HOST_MAC enumeration member 933
POCA_PAUSE enumeration member 1033
POCA_RESTART enumeration member 1033
POCM_AUTO enumeration member 1014
POCM_DISABLED enumeration member 1014
POCM_ENABLED enumeration member 1014
POCO_DUMP enumeration member 1033
POCO_FULL enumeration member 1033
POCO_REPORT enumeration member 1033
POD_OPTIMIZE_FOR_ACCESSIBILITY enumeration member 1015
POD_OPTIMIZE_FOR_GAMES enumeration member 1015
POD_STANDARD enumeration member 1015
PPF_TCP enumeration member 1015
PPF_UDP enumeration member 1015
PPRF_ADD_CLIENT_PART enumeration member 1018
PPRF_ADD_SERVER_PART enumeration member 1018
PPRF_DO_NOT_CREATE_HOST_SCREENSHOT enumeration member 1018
PPS_PROC_IDLE enumeration member 1020
PPS_PROC_RUN enumeration member 1020
PPS_PROC_SLEEP enumeration member 1020
PPS_PROC_STOP enumeration member 1020
PPS_PROC_ZOMBIE enumeration member 1020
PR3F_ALLOW_UNKNOWN_OS enumeration member 1022
PR3F_CREATE_SHADOW_VM enumeration member 1022
PRCF_FORCE enumeration member 1022
PRCF_UNREG_PRESERVE enumeration member 1022
PRD_AUTO enumeration member 934
PRD_DISABLED enumeration member 934
PRD_MANUAL enumeration member 934
PRIF_DISK_INFO enumeration member 1023
PRIF_RESIZE_LAST_PARTITION enumeration member 1023
PRIF_RESIZE_OFFLINE enumeration member 1023
Primary Display Capture 778
PRL_ADAPTER_INDEX 1036
PRL_ADAPTER_INDEX enumeration 1036
PRL_ALLOWED_VM_COMMAND 1036
PRL_ALLOWED_VM_COMMAND enumeration 1036
PRL_ALLOWED_VM_COMMAND_PTR 812
PRL_ALLOWED_VM_COMMAND_PTR type 812
PRL_API_COMMAND_FLAGS 1037
PRL_API_COMMAND_FLAGS enumeration 1037
PRL_API_COMMAND_FLAGS_PTR 812

PRL_API_COMMAND_FLAGS_PTR type 812
PRL_API_INIT_FLAGS 1038
PRL_API_INIT_FLAGS enumeration 1038
PRL_API_INIT_FLAGS_PTR 812
PRL_API_INIT_FLAGS_PTR type 812
PRL_APP_IN_DOCK_MODE 1038
PRL_APP_IN_DOCK_MODE enumeration 1038
PRL_APP_IN_DOCK_MODE_PTR 812
PRL_APP_IN_DOCK_MODE_PTR type 812
PRL_APPLIANCE_DOWNLOAD_STATUS 1039
PRL_APPLIANCE_DOWNLOAD_STATUS enumeration 1039
PRL_APPLIANCE_DOWNLOAD_STATUS_PTR 812
PRL_APPLIANCE_DOWNLOAD_STATUS_PTR type 812
PRL_APPLICATION_MODE 1039
PRL_APPLICATION_MODE enumeration 1039
PRL_APPLICATION_MODE_PTR 813
PRL_APPLICATION_MODE_PTR type 813
PRL_ASYNC_METHOD 1622
PRL_ASYNC_METHOD macro 1622
PRL_ASYNC_METHOD_PTR 1622
PRL_ASYNC_METHOD_PTR macro 1622
PRL_BEGIN_VM_BACKUP_FLAGS 1039
PRL_BEGIN_VM_BACKUP_FLAGS enumeration 1039
PRL_BIOS_TYPE 1040
PRL_BIOS_TYPE enumeration 1040
PRL_BIOS_TYPE_PTR 813
PRL_BIOS_TYPE_PTR type 813
PRL_BOOL 782
PRL_BOOL type 782
PRL_BOOL Types 782
PRL_BOOL_PTR 783
PRL_BOOL_PTR type 783
PRL_BSD_PARTITION 1622
PRL_BSD_PARTITION macro 1622
PRL_CALL 1623
PRL_CALL macro 1623
PRL_CHAR 783
PRL_CHAR type 783
PRL_CHAR Types 783
PRL_CHAR_PTR 783
PRL_CHAR_PTR type 783
PRL_CHECKED_DISK_INVALID 1623
PRL_CHECKED_DISK_INVALID macro 1623
PRL_CHECKED_DISK_OLD_VERSION 1623
PRL_CHECKED_DISK_OLD_VERSION macro 1623
PRL_CHECKED_DISK_VALID 1623
PRL_CHECKED_DISK_VALID macro 1623
PRL_CHS 1040
PRL_CHS structure 1040
PRL_CHS_PTR 813
PRL_CHS_PTR type 813
PRL_CLONE_VM_FLAGS 1040
PRL_CLONE_VM_FLAGS enumeration 1040
PRL_CLONE_VM_FLAGS_PTR 813
PRL_CLONE_VM_FLAGS_PTR type 813
PRL_CLUSTERED_DEVICE_SUBTYPE 1041
PRL_CLUSTERED_DEVICE_SUBTYPE enumeration 1041
PRL_CLUSTERED_DEVICE_SUBTYPE_PTR 814
PRL_CLUSTERED_DEVICE_SUBTYPE_PTR type 814
PRL_COMMIT_ENCRYPTION_FLAGS 1041
PRL_COMMIT_ENCRYPTION_FLAGS enumeration 1041
PRL_COMMIT_ENCRYPTION_FLAGS_PTR 814
PRL_COMMIT_ENCRYPTION_FLAGS_PTR type 814
PRL_COMPACT_MODE 1041
PRL_COMPACT_MODE enumeration 1041
PRL_COMPACT_MODE_PTR 814
PRL_COMPACT_MODE_PTR type 814
PRL_CONST Types 784
PRL_CONST_BOOL_PTR 784
PRL_CONST_BOOL_PTR type 784
PRL_CONST_DISK_OPEN_FLAGS_PTR 784
PRL_CONST_DISK_OPEN_FLAGS_PTR type 784
PRL_CONST_DISK_PARAMETERS_PTR 785
PRL_CONST_DISK_PARAMETERS_PTR type 785
PRL_CONST_DOUBLE_PTR 785
PRL_CONST_DOUBLE_PTR type 785
PRL_CONST_HANDLE_PTR 785
PRL_CONST_HANDLE_PTR type 785

PRL_CONST_INT16_PTR 785
PRL_CONST_INT16_PTR type 785
PRL_CONST_INT32_PTR 785
PRL_CONST_INT32_PTR type 785
PRL_CONST_INT64_PTR 786
PRL_CONST_INT64_PTR type 786
PRL_CONST_INT8_PTR 786
PRL_CONST_INT8_PTR type 786
PRL_CONST_SIZE_PTR 786
PRL_CONST_SIZE_PTR type 786
PRL_CONST_SNAPSHOT_ELEMENT_PTR 786
PRL_CONST_SNAPSHOT_ELEMENT_PTR type 786
PRL_CONST_STR 786
PRL_CONST_STR type 786
PRL_CONST_STR_PTR 787
PRL_CONST_STR_PTR type 787
PRL_CONST_UINT16_PTR 787
PRL_CONST_UINT16_PTR type 787
PRL_CONST_UINT32_PTR 787
PRL_CONST_UINT32_PTR type 787
PRL_CONST_UINT64_PTR 787
PRL_CONST_UINT64_PTR type 787
PRL_CONST_UINT8_PTR 787
PRL_CONST_UINT8_PTR type 787
PRL_CONST_VOID_PTR 788
PRL_CONST_VOID_PTR type 788
PRL_CONVERT_VIRTUAL_DISK 1042
PRL_CONVERT_VIRTUAL_DISK enumeration 1042
PRL_COPY_CT_TEMPLATE_FLAGS 1042
PRL_COPY_CT_TEMPLATE_FLAGS enumeration 1042
PRL_CORNER_ACTIONS 1043
PRL_CORNER_ACTIONS enumeration 1043
PRL_CORNER_ACTIONS_PTR 814
PRL_CORNER_ACTIONS_PTR type 814
PRL_CPU_FEATURES_EX 1043
PRL_CPU_FEATURES_EX enumeration 1043
PRL_CPU_HVT 1043
PRL_CPU_HVT enumeration 1043
PRL_CPU_HVT_PTR 814
PRL_CPU_HVT_PTR type 814
PRL_CPU_MODE 1044
PRL_CPU_MODE enumeration 1044
PRL_CPU_MODE_PTR 815
PRL_CPU_MODE_PTR type 815
PRL_CPU_UNLIMITED enumeration member 977
PRL_CPU_VTX_MODE 815
PRL_CPU_VTX_MODE type 815
PRL_CPU_VTX_MODE_PTR 815
PRL_CPU_VTX_MODE_PTR type 815
PRL_CPULIMIT_MHZ enumeration member 977
PRL_CPULIMIT_PERCENTS enumeration member 977
PRL_CPULIMIT_PERCENTS_TO_MHZ enumeration member 977
PRL_CPULIMIT_PERCENTS100 enumeration member 977
PRL_CPULIMIT_TYPE 1044
PRL_CPULIMIT_TYPE enumeration 1044
PRL_CPUS_COUNT 1044
PRL_CPUS_COUNT enumeration 1044
PRL_CPUUITS 1045
PRL_CPUUITS enumeration 1045
PRL_CPUUNITS_DEFAULT enumeration member 978
PRL_CPUUNITS_MAX enumeration member 978
PRL_CPUUNITS_MIN enumeration member 978
PRL_CREATE_DISK_IMAGE_FLAGS 1045
PRL_CREATE_DISK_IMAGE_FLAGS enumeration 1045
PRL_CREATE_DISK_IMAGE_FLAGS_PTR 815
PRL_CREATE_DISK_IMAGE_FLAGS_PTR type 815
PRL_CREATE_SNAPSHOT_FLAGS 1045
PRL_CREATE_SNAPSHOT_FLAGS enumeration 1045
PRL_CT_CAPMASK 1046
PRL_CT_CAPMASK enumeration 1046
PRL_CT_CAPMASK_PTR 815
PRL_CT_CAPMASK_PTR type 815
PRL_CT_FEATURES 1047
PRL_CT_FEATURES enumeration 1047
PRL_CT_REINSTALL_OPTIONS 1047
PRL_CT_REINSTALL_OPTIONS enumeration 1047
PRL_CT_REINSTALL_OPTIONS_PTR 816
PRL_CT_REINSTALL_OPTIONS_PTR type 816

PRL_CT_RESOURCE 1047
PRL_CT_RESOURCE enumeration 1047
PRL_CT_TEMPLATE_TYPE 1048
PRL_CT_TEMPLATE_TYPE enumeration 1048
PRL_CT_TEMPLATE_TYPE_PTR 816
PRL_CT_TEMPLATE_TYPE_PTR type 816
PRL_CURRENT_GUEST_OS_SESSION 1624
PRL_CURRENT_GUEST_OS_SESSION macro 1624
PRL_DATA_PARTITION 1624
PRL_DATA_PARTITION macro 1624
PRL_DATA_STATISTIC_SEGMENTS 1048
PRL_DATA_STATISTIC_SEGMENTS enumeration 1048
PRL_DATA_STATISTIC_SEGMENTS_PTR 816
PRL_DATA_STATISTIC_SEGMENTS_PTR type 816
PRL_DELETE_SNAPSHOT_FLAGS 1048
PRL_DELETE_SNAPSHOT_FLAGS enumeration 1048
PRL_DEVICE Types 788
PRL_DEVICE_ACTION 1049
PRL_DEVICE_ACTION enumeration 1049
PRL_DEVICE_ACTION_PTR 788
PRL_DEVICE_ACTION_PTR type 788
PRL_DEVICE_TYPE 1049
PRL_DEVICE_TYPE enumeration 1049
PRL_DEVICE_TYPE_PTR 788
PRL_DEVICE_TYPE_PTR type 788
PRL_DISK Macros 1128
PRL_DISK Types 788
PRL_DISK_ALIGN_REQUESTS 1129
PRL_DISK_ALIGN_REQUESTS macro 1129
PRL_DISK_ALLOW_DATA_MODIFY 1129
PRL_DISK_ALLOW_DATA_MODIFY macro 1129
PRL_DISK_CACHE_DEFAULT 1130
PRL_DISK_CACHE_DEFAULT macro 1130
PRL_DISK_CACHE_DISABLED 1130
PRL_DISK_CACHE_DISABLED macro 1130
PRL_DISK_CACHE_ENABLED 1130
PRL_DISK_CACHE_ENABLED macro 1130
PRL_DISK_CACHE_UNCACHED_READ 1130
PRL_DISK_CACHE_UNCACHED_READ macro 1130
PRL_DISK_CACHE_UNCACHED_SEQ_READ 1131
PRL_DISK_CACHE_UNCACHED_SEQ_READ macro 1131
PRL_DISK_CALLBACK 779
PRL_DISK_CALLBACK type 779
PRL_DISK_DEFAULT_FLAG 1131
PRL_DISK_DEFAULT_FLAG macro 1131
PRL_DISK_FAKE_OPEN 1131
PRL_DISK_FAKE_OPEN macro 1131
PRL_DISK_FOPS_DEFAULT 1132
PRL_DISK_FOPS_DEFAULT macro 1132
PRL_DISK_FOPS_MASK 1132
PRL_DISK_FOPS_MASK macro 1132
PRL_DISK_FOPS_MERGING 1132
PRL_DISK_FOPS_MERGING macro 1132
PRL_DISK_FOPS_NONCACHED 1132
PRL_DISK_FOPS_NONCACHED macro 1132
PRL_DISK_FOPS_NORMAL 1133
PRL_DISK_FOPS_NORMAL macro 1133
PRL_DISK_FOPS_PCS 1133
PRL_DISK_FOPS_PCS macro 1133
PRL_DISK_IGNORE_PROTECTION 1133
PRL_DISK_IGNORE_PROTECTION macro 1133
PRL_DISK_INVALID 1133
PRL_DISK_INVALID macro 1133
PRL_DISK_NO_ERROR_CHECKING 1134
PRL_DISK_NO_ERROR_CHECKING macro 1134
PRL_DISK_NO_IMAGE_CREATION 1624
PRL_DISK_NO_IMAGE_CREATION macro 1624
PRL_DISK_NO_SPACE_CHECKS 1134
PRL_DISK_NO_SPACE_CHECKS macro 1134
PRL_DISK_NOT_LIMIT_HANDLES 1134
PRL_DISK_NOT_LIMIT_HANDLES macro 1134
PRL_DISK_NOTHING 1134
PRL_DISK_NOTHING macro 1134
PRL_DISK_OPEN_FLAGS 789
PRL_DISK_OPEN_FLAGS type 789
PRL_DISK_OPEN_FLAGS_PTR 789
PRL_DISK_OPEN_FLAGS_PTR type 789
PRL_DISK_PARAMETERS 1050

PRL_DISK_PARAMETERS structure 1050
PRL_DISK_PARAMETERS_PTR 789
PRL_DISK_PARAMETERS_PTR type 789
PRL_DISK_PROGRESS_COMPLETED 1135
PRL_DISK_PROGRESS_COMPLETED macro 1135
PRL_DISK_PROGRESS_MAX 1135
PRL_DISK_PROGRESS_MAX macro 1135
PRL_DISK_PROGRESS_MIN 1135
PRL_DISK_PROGRESS_MIN macro 1135
PRL_DISK_READ 1135
PRL_DISK_READ macro 1135
PRL_DISK_READ_LOCK 1135
PRL_DISK_READ_LOCK macro 1135
PRL_DISK_USE_RAW_DISK 1136
PRL_DISK_USE_RAW_DISK macro 1136
PRL_DISK_WRITE 1136
PRL_DISK_WRITE macro 1136
PRL_DISK_XML_CHANGE 1136
PRL_DISK_XML_CHANGE macro 1136
PRL_DISPATCHER_LISTEN_PORT 1624
PRL_DISPATCHER_LISTEN_PORT macro 1624
PRL_DISPATCHER_LISTEN_PORT_PMOBILE 1624
PRL_DISPATCHER_LISTEN_PORT_PMOBILE macro 1624
PRL_DISPLAY_BUFFER_FORMAT 1050
PRL_DISPLAY_BUFFER_FORMAT enumeration 1050
PRL_DISPLAY_BUFFER_FORMAT_PTR 816
PRL_DISPLAY_BUFFER_FORMAT_PTR type 816
PRL_DISPLAY_CODEC_COMPRESSION 1051
PRL_DISPLAY_CODEC_COMPRESSION enumeration 1051
PRL_DISPLAY_CODEC_QUALITY 1051
PRL_DISPLAY_CODEC_QUALITY enumeration 1051
PRL_DISPLAY_CODEC_TYPE 1051
PRL_DISPLAY_CODEC_TYPE enumeration 1051
PRL_DISPLAY_CONFIG_MODE 1052
PRL_DISPLAY_CONFIG_MODE enumeration 1052
PRL_DISPLAY_CONFIG_MODE_PTR 816
PRL_DISPLAY_CONFIG_MODE_PTR type 816
PRL_DISTRO_MEDIA_TYPE 1052
PRL_DISTRO_MEDIA_TYPE enumeration 1052
PRL_DISTRO_MEDIA_TYPE_PTR 817
PRL_DISTRO_MEDIA_TYPE_PTR type 817
PRL_DOCK_ICON_TYPE 1053
PRL_DOCK_ICON_TYPE enumeration 1053
PRL_DOCK_ICON_TYPE_PTR 817
PRL_DOCK_ICON_TYPE_PTR type 817
PRL_DOUBLE 790
PRL_DOUBLE type 790
PRL_DOUBLE Types 789
PRL_DOUBLE_PTR 790
PRL_DOUBLE_PTR type 790
PRL_DOUBLE_PTR_PTR 790
PRL_DOUBLE_PTR_PTR type 790
PRL_EFI_PARTITION 1625
PRL_EFI_PARTITION macro 1625
PRL_EFI_PARTITION_GPT 1625
PRL_EFI_PARTITION_GPT macro 1625
PRL_EMPTY_PARTITION 1625
PRL_EMPTY_PARTITION macro 1625
PRL_END_VM_BACKUP_FLAGS 1053
PRL_END_VM_BACKUP_FLAGS enumeration 1053
PRL_ERR Macros 1137
PRL_ERR_ACCES_DENIED_FILE_TO_PARENT_PARENT_DIR 1175
PRL_ERR_ACCES_DENIED_FILE_TO_PARENT_PARENT_DIR macro 1175
PRL_ERR_ACCESS_DENIED 1175
PRL_ERR_ACCESS_DENIED macro 1175
PRL_ERR_ACCESS_DENIED_TO_CHANGE_PERMISSIONS 1175
PRL_ERR_ACCESS_DENIED_TO_CHANGE_PERMISSIONS macro 1175
PRL_ERR_ACCESS_DENIED_TO_DISK_IMAGE 1176
PRL_ERR_ACCESS_DENIED_TO_DISK_IMAGE macro 1176
PRL_ERR_ACCESS_DENINED_TO_RUN_WIZARD 1176
PRL_ERR_ACCESS_DENINED_TO_RUN_WIZARD macro 1176
PRL_ERR_ACCESS_TO_CLONE_VM_DEVICE_DENIED 1176
PRL_ERR_ACCESS_TO_CLONE_VM_DEVICE_DENIED macro 1176

PRL_ERR_ACCESS_TO_VM_DENIED 1176

PRL_ERR_ACCESS_TO_VM_DENIED macro 1176

PRL_ERR_ACCESS_TO_VM_HDD_DENIED 1176

PRL_ERR_ACCESS_TO_VM_HDD_DENIED macro 1176

PRL_ERR_ACCESS_TOKEN_INVALID 1177

PRL_ERR_ACCESS_TOKEN_INVALID macro 1177

PRL_ERR_ACTION_NOT_SUPPORTED_FOR_CT 1177

PRL_ERR_ACTION_NOT_SUPPORTED_FOR_CT macro 1177

PRL_ERR_ACTIVATE_NO_INSTALLED_LICENSE 1177

PRL_ERR_ACTIVATE_NO_INSTALLED_LICENSE macro 1177

PRL_ERR_ACTIVATE_TRIAL_LICENSE 1177

PRL_ERR_ACTIVATE_TRIAL_LICENSE macro 1177

PRL_ERR_ACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE
1178

PRL_ERR_ACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE
macro 1178

PRL_ERR_ACTIVATION_COMMON_SERVER_ERROR 1178

PRL_ERR_ACTIVATION_COMMON_SERVER_ERROR macro 1178

PRL_ERR_ACTIVATION_HTTP_REQUEST_FAILED 1178

PRL_ERR_ACTIVATION_HTTP_REQUEST_FAILED macro 1178

PRL_ERR_ACTIVATION_OFFLINE_PERIOD_EXPIRED 1178

PRL_ERR_ACTIVATION_OFFLINE_PERIOD_EXPIRED macro 1178

PRL_ERR_ACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID
1179

PRL_ERR_ACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID
macro 1179

PRL_ERR_ACTIVATION_SERVER_ERROR 1179

PRL_ERR_ACTIVATION_SERVER_ERROR macro 1179

PRL_ERR_ACTIVATION_SERVER_HWIDS_AMOUNT_REACHED
1179

PRL_ERR_ACTIVATION_SERVER_HWIDS_AMOUNT_REACHED
macro 1179

PRL_ERR_ACTIVATION_SERVER_KEY_IS_BLACKLISTED
1179

PRL_ERR_ACTIVATION_SERVER_KEY_IS_BLACKLISTED
macro 1179

PRL_ERR_ACTIVATION_SERVER_KEY_IS_INVALID 1180

PRL_ERR_ACTIVATION_SERVER_KEY_IS_INVALID macro 1180

PRL_ERR_ACTIVATION_UNABLE_TO_SEND_REQUEST 1180

PRL_ERR_ACTIVATION_UNABLE_TO_SEND_REQUEST macro 1180

PRL_ERR_ACTIVATION_UPDATE_FAILED 1180

PRL_ERR_ACTIVATION_UPDATE_FAILED macro 1180

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_FORMAT
1180

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_FORMAT
macro 1180

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_MESSAGE
1181

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_MESSAGE
macro 1181

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_SIGNATURE
1181

PRL_ERR_ACTIVATION_WRONG_CONFIRMATION_SIGNATURE
macro 1181

PRL_ERR_ACTIVATION_WRONG_SERVER_RESPONSE 1181

PRL_ERR_ACTIVATION_WRONG_SERVER_RESPONSE macro 1181

PRL_ERR_ADD_HW_FLOPPY_IMAGE_NOT_SPECIFYED 1181

PRL_ERR_ADD_HW_FLOPPY_IMAGE_NOT_SPECIFYED macro 1181

PRL_ERR_ADD_VM_OPERATION_WAS_CANCELED 1182

PRL_ERR_ADD_VM_OPERATION_WAS_CANCELED macro 1182

PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_OPERATION
1182

PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_OPERATION
macro 1182

PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_VM_OPERATION
1182

PRL_ERR_ADMIN_CONFIRMATION_IS_REQUIRED_FOR_VM_OPERATION

macro 1182

PRL_ERR_ALIGN_ON_4 1182

PRL_ERR_ALIGN_ON_4 macro 1182

PRL_ERR_ALREADY_CONNECTED_TO_DISPATCHER 1183

PRL_ERR_ALREADY_CONNECTED_TO_DISPATCHER macro 1183

PRL_ERR_ANOTHER_USER_SESSIONS_PRESENT 1183

PRL_ERR_ANOTHER_USER_SESSIONS_PRESENT macro 1183

PRL_ERR_API_INCOMPATIBLE 1183

PRL_ERR_API_INCOMPATIBLE macro 1183

PRL_ERR_API_WASNT_INITIALIZED 1183

PRL_ERR_API_WASNT_INITIALIZED macro 1183

PRL_ERR_APPLIANCE_CANNOT_CALCULATE_MD5 1183

PRL_ERR_APPLIANCE_CANNOT_CALCULATE_MD5 macro 1183

PRL_ERR_APPLIANCE_CANNOT_CHANGE_OWNER 1184

PRL_ERR_APPLIANCE_CANNOT_CHANGE_OWNER macro 1184

PRL_ERR_APPLIANCE_CANNOT_EXTRACT_VM 1184

PRL_ERR_APPLIANCE_CANNOT_EXTRACT_VM macro 1184

PRL_ERR_APPLIANCE_CANNOT_MOVE_VM_BUNDLE 1184

PRL_ERR_APPLIANCE_CANNOT_MOVE_VM_BUNDLE macro 1184

PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_DIR 1184

PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_DIR macro 1184

PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_EXISTS 1185

PRL_ERR_APPLIANCE_DOWNLOAD_PARENT_PATH_NOT_EXISTS macro 1185

PRL_ERR_APPLIANCE_DOWNLOAD_PATH_CANNOT_CREATE 1185

PRL_ERR_APPLIANCE_DOWNLOAD_PATH_CANNOT_CREATE macro 1185

PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_FOUND 1185

PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_FOUND macro 1185

PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_STARTED 1185

PRL_ERR_APPLIANCE_DOWNLOAD_UTILITY_NOT_STARTED macro 1185

PRL_ERR_APPLIANCE_EXIT_WITH_ERROR 1186

PRL_ERR_APPLIANCE_EXIT_WITH_ERROR macro 1186

PRL_ERR_APPLIANCE_INSTALL_ALREADY_IN_PROCESS 1186

PRL_ERR_APPLIANCE_INSTALL_ALREADY_IN_PROCESS macro 1186

PRL_ERR_APPLIANCE_INSTALL_WAS_NOT_STARTED 1186

PRL_ERR_APPLIANCE_INSTALL_WAS_NOT_STARTED macro 1186

PRL_ERR_APPLIANCE_INVALID_CONFIG 1186

PRL_ERR_APPLIANCE_INVALID_CONFIG macro 1186

PRL_ERR_APPLIANCE_MISMATCH_MD5 1187

PRL_ERR_APPLIANCE_MISMATCH_MD5 macro 1187

PRL_ERR_APPLIANCE_NAME_NOT_MATCH_VM_BUNDLE 1187

PRL_ERR_APPLIANCE_NAME_NOT_MATCH_VM_BUNDLE macro 1187

PRL_ERR_APPLIANCE_USER_NOT_FOUND 1187

PRL_ERR_APPLIANCE_USER_NOT_FOUND macro 1187

PRL_ERR_ATTACH_BACKUP_ALREADY_ATTACHED 1187

PRL_ERR_ATTACH_BACKUP_ALREADY_ATTACHED macro 1187

PRL_ERR_ATTACH_BACKUP_BUSE_NOT_MOUNTED 1187

PRL_ERR_ATTACH_BACKUP_BUSE_NOT_MOUNTED macro 1187

PRL_ERR_ATTACH_BACKUP_CUSTOM_BACKUP_SERVER_NOT_SUPPORTED 1188

PRL_ERR_ATTACH_BACKUP_CUSTOM_BACKUP_SERVER_NOT_SUPPORTED macro 1188

PRL_ERR_ATTACH_BACKUP_FORMAT_NOT_SUPPORTED 1188

PRL_ERR_ATTACH_BACKUP_FORMAT_NOT_SUPPORTED macro 1188

PRL_ERR_ATTACH_BACKUP_INTERNAL_ERROR 1188

PRL_ERR_ATTACH_BACKUP_INTERNAL_ERROR macro 1188

PRL_ERR_ATTACH_BACKUP_INVALID_STORAGE_URL 1188

PRL_ERR_ATTACH_BACKUP_INVALID_STORAGE_URL macro 1188

PRL_ERR_ATTACH_BACKUP_PROTO_ERROR 1189

PRL_ERR_ATTACH_BACKUP_PROTO_ERROR macro 1189

PRL_ERR_ATTACH_BACKUP_URL_CHANGE_PROHIBITED 1189

PRL_ERR_ATTACH_BACKUP_URL_CHANGE_PROHIBITED macro 1189

PRL_ERR_ATTACH_TO_TASK_BAD_SESSION 1189

PRL_ERR_ATTACH_TO_TASK_BAD_SESSION macro 1189

PRL_ERR_AUDIO_ENCODINGS_NOT_SUPPORTED 1189

PRL_ERR_AUDIO_ENCODINGS_NOT_SUPPORTED macro 1189

PRL_ERR_AUTHENTICATION_FAILED 1190

PRL_ERR_AUTHENTICATION_FAILED macro 1190

PRL_ERR_BACKUP_ACCESS_TO_VM_DENIED 1190

PRL_ERR_BACKUP_ACCESS_TO_VM_DENIED macro 1190

PRL_ERR_BACKUP_ACRONIS_ERR 1190

PRL_ERR_BACKUP_ACRONIS_ERR macro 1190

PRL_ERR_BACKUP_BACKUP_CMD_FAILED 1190

PRL_ERR_BACKUP_BACKUP_CMD_FAILED macro 1190

PRL_ERR_BACKUP_BACKUP_NOT_FOUND 1190

PRL_ERR_BACKUP_BACKUP_NOT_FOUND macro 1190

PRL_ERR_BACKUP_BACKUP_UUID_NOT_FOUND 1191

PRL_ERR_BACKUP_BACKUP_UUID_NOT_FOUND macro 1191

PRL_ERR_BACKUP_CANNOT_CREATE_DIRECTORY 1191

PRL_ERR_BACKUP_CANNOT_CREATE_DIRECTORY macro 1191

PRL_ERR_BACKUP_CANNOT_REMOVE_DIRECTORY 1191

PRL_ERR_BACKUP_CANNOT_REMOVE_DIRECTORY macro 1191

PRL_ERR_BACKUP_CANNOT_SET_PERMISSIONS 1191

PRL_ERR_BACKUP_CANNOT_SET_PERMISSIONS macro 1191

PRL_ERR_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE 1192

PRL_ERR_BACKUP_CREATE_NOT_ENOUGH_FREE_DISK_SPACE macro 1192

PRL_ERR_BACKUP_CREATE_SNAPSHOT_FAILED 1192

PRL_ERR_BACKUP_CREATE_SNAPSHOT_FAILED macro 1192

PRL_ERR_BACKUP_CT_ID_ALREADY_EXIST 1192

PRL_ERR_BACKUP_CT_ID_ALREADY_EXIST macro 1192

PRL_ERR_BACKUP_DIRECTORY_ALREADY_EXIST 1192

PRL_ERR_BACKUP_DIRECTORY_ALREADY_EXIST macro 1192

PRL_ERR_BACKUP_INTERNAL_ERROR 1193

PRL_ERR_BACKUP_INTERNAL_ERROR macro 1193

PRL_ERR_BACKUP_INTERNAL_PROTO_ERROR 1193

PRL_ERR_BACKUP_INTERNAL_PROTO_ERROR macro 1193

PRL_ERR_BACKUP_LOCKED_FOR_READING 1193

PRL_ERR_BACKUP_LOCKED_FOR_READING macro 1193

PRL_ERR_BACKUP_LOCKED_FOR_WRITING 1193

PRL_ERR_BACKUP_LOCKED_FOR_WRITING macro 1193

PRL_ERR_BACKUP_REGISTER_VM_FAILED 1193

PRL_ERR_BACKUP_REGISTER_VM_FAILED macro 1193

PRL_ERR_BACKUP_REMOVE_PERMISSIONS_DENIED 1194

PRL_ERR_BACKUP_REMOVE_PERMISSIONS_DENIED macro 1194

PRL_ERR_BACKUP_REQUIRE_LOGIN_PASSWORD 1194

PRL_ERR_BACKUP_REQUIRE_LOGIN_PASSWORD macro 1194

PRL_ERR_BACKUP_RESTORE_CANNOT_CREATE_DIRECTORY 1194

PRL_ERR_BACKUP_RESTORE_CANNOT_CREATE_DIRECTORY macro 1194

PRL_ERR_BACKUP_RESTORE_CMD_FAILED 1194

PRL_ERR_BACKUP_RESTORE_CMD_FAILED macro 1194

PRL_ERR_BACKUP_RESTORE_DIRECTORY_ALREADY_EXIST 1195

PRL_ERR_BACKUP_RESTORE_DIRECTORY_ALREADY_EXIST macro 1195

PRL_ERR_BACKUP_RESTORE_EXISTING_SET_PRIVATE 1195

PRL_ERR_BACKUP_RESTORE_EXISTING_SET_PRIVATE macro 1195

PRL_ERR_BACKUP_RESTORE_INTERNAL_ERROR 1195

PRL_ERR_BACKUP_RESTORE_INTERNAL_ERROR macro 1195

PRL_ERR_BACKUP_RESTORE_INTERNAL_PROTO_ERROR 1195

PRL_ERR_BACKUP_RESTORE_INTERNAL_PROTO_ERROR
macro 1195
PRL_ERR_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE
1196
PRL_ERR_BACKUP_RESTORE_NOT_ENOUGH_FREE_DISK_SPACE
macro 1196
PRL_ERR_BACKUP_RESTORE_PROHIBIT_WHEN_ATTACHED
1196
PRL_ERR_BACKUP_RESTORE_PROHIBIT_WHEN_ATTACHED
macro 1196
PRL_ERR_BACKUP_RESTORE_VM_RUNNING 1196
PRL_ERR_BACKUP_RESTORE_VM_RUNNING macro 1196
PRL_ERR_BACKUP_SNAPSHOT_OF_PAUSED_VM 1196
PRL_ERR_BACKUP_SNAPSHOT_OF_PAUSED_VM macro
1196
PRL_ERR_BACKUP_SWITCH_TO_SNAPSHOT_FAILED 1197
PRL_ERR_BACKUP_SWITCH_TO_SNAPSHOT_FAILED
macro 1197
PRL_ERR_BACKUP_TIMEOUT_EXCEEDED 1197
PRL_ERR_BACKUP_TIMEOUT_EXCEEDED macro 1197
PRL_ERR_BACKUP_TOOL_CANNOT_START 1197
PRL_ERR_BACKUP_TOOL_CANNOT_START macro 1197
PRL_ERR_BACKUP_UNSUPPORTED_KEEP_CHAIN 1197
PRL_ERR_BACKUP_UNSUPPORTED_KEEP_CHAIN macro
1197
PRL_ERR_BACKUP_UNSUPPORTED_STORAGE_VERSION
1198
PRL_ERR_BACKUP_UNSUPPORTED_STORAGE_VERSION
macro 1198
PRL_ERR_BAD_DISP_CONFIG_FILE_SPECIFIED 1198
PRL_ERR_BAD_DISP_CONFIG_FILE_SPECIFIED macro 1198
PRL_ERR_BAD_PARAMETERS 1198
PRL_ERR_BAD_PARAMETERS macro 1198
PRL_ERR_BAD_VM_CONFIG_FILE_SPECIFIED 1198
PRL_ERR_BAD_VM_CONFIG_FILE_SPECIFIED macro 1198
PRL_ERR_BAD_VM_DIR_CONFIG_FILE_SPECIFIED 1198
PRL_ERR_BAD_VM_DIR_CONFIG_FILE_SPECIFIED macro
1198
PRL_ERR_BRIDGE_NOT_FOUND_FOR_NETWORK_ADAPTER
1199
PRL_ERR_BRIDGE_NOT_FOUND_FOR_NETWORK_ADAPTER
macro 1199
PRL_ERR_BUFFER_OVERRUN 1199
PRL_ERR_BUFFER_OVERRUN macro 1199
PRL_ERR_BUSE_ENTRY_ALREADY_EXIST 1199
PRL_ERR_BUSE_ENTRY_ALREADY_EXIST macro 1199
PRL_ERR_BUSE_ENTRY_ALREADY_INITIALIZED 1199
PRL_ERR_BUSE_ENTRY_ALREADY_INITIALIZED macro
1199
PRL_ERR_BUSE_ENTRY_INVALID 1200
PRL_ERR_BUSE_ENTRY_INVALID macro 1200
PRL_ERR_BUSE_ENTRY_IO_ERROR 1200
PRL_ERR_BUSE_ENTRY_IO_ERROR macro 1200
PRL_ERR_BUSE_INTERNAL_ERROR 1200
PRL_ERR_BUSE_INTERNAL_ERROR macro 1200
PRL_ERR_BUSE_NOT_MOUNTED 1200
PRL_ERR_BUSE_NOT_MOUNTED macro 1200
PRL_ERR_CAN_NOT_GET_DISK_FREE_SPACE 1200
PRL_ERR_CAN_NOT_GET_DISK_FREE_SPACE macro 1200
PRL_ERR_CANNOT_EDIT_FIREWALL_AT_TRANS_VM_STATE
1201
PRL_ERR_CANNOT_EDIT_FIREWALL_AT_TRANS_VM_STATE
macro 1201
PRL_ERR_CANNOT_EDIT_HARDWARE_FOR_PAUSED_VM
1201
PRL_ERR_CANNOT_EDIT_HARDWARE_FOR_PAUSED_VM
macro 1201
PRL_ERR_CANNOT_PROCESSING_SAFE_MODE_FOR_INVALID_VM
1201
PRL_ERR_CANNOT_PROCESSING_SAFE_MODE_FOR_INVALID_VM
macro 1201
PRL_ERR_CANNOT_PROCESSING_UNDO_DISKS_FOR_INVALID_VM
1202
PRL_ERR_CANNOT_PROCESSING_UNDO_DISKS_FOR_INVALID_VM
macro 1202
PRL_ERR_CANNOT_SAVE_REMOTE_DEVICE_STATE 1202
PRL_ERR_CANNOT_SAVE_REMOTE_DEVICE_STATE
macro 1202
PRL_ERR_CANT_ALOCA_MEM_FILE_COPY 1202

PRL_ERR_CANT_ALOCA_MEM_FILE_COPY macro 1202
PRL_ERR_CANT_CHANGE_FILE_PERMISSIONS 1202
PRL_ERR_CANT_CHANGE_FILE_PERMISSIONS macro 1202
PRL_ERR_CANT_CHANGE_HOST_ID 1203
PRL_ERR_CANT_CHANGE_HOST_ID macro 1203
PRL_ERR_CANT_CHANGE_OWNER_OF_DISK_IMAGE_FILE 1203
PRL_ERR_CANT_CHANGE_OWNER_OF_DISK_IMAGE_FILE macro 1203
PRL_ERR_CANT_CHANGE_OWNER_OF_FILE 1203
PRL_ERR_CANT_CHANGE_OWNER_OF_FILE macro 1203
PRL_ERR_CANT_CHANGE_OWNER_OF_VM_FILE 1203
PRL_ERR_CANT_CHANGE_OWNER_OF_VM_FILE macro 1203
PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL 1203
PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL macro 1203
PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL_BY_NON_PRIVILEGED_USER 1204
PRL_ERR_CANT_CHANGE_PROXY_MANAGER_URL_BY_NON_PRIVILEGED_USER macro 1204
PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN 1204
PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN macro 1204
PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN_BY_NON_PRIVILEGED_USER 1204
PRL_ERR_CANT_CHANGE_WEB_PORTAL_DOMAIN_BY_NON_PRIVILEGED_USER macro 1204
PRL_ERR_CANT_CONFIGURE_PARTITION_HDD 1204
PRL_ERR_CANT_CONFIGURE_PARTITION_HDD macro 1204
PRL_ERR_CANT_CONFIGURE_PHYSICAL_HDD 1205
PRL_ERR_CANT_CONFIGURE_PHYSICAL_HDD macro 1205
PRL_ERR_CANT_CONNECT_TO_DISPATCHER 1205
PRL_ERR_CANT_CONNECT_TO_DISPATCHER macro 1205
PRL_ERR_CANT_CONNECT_TO_DISPATCHER_ITERATIVELY 1205
PRL_ERR_CANT_CONNECT_TO_DISPATCHER_ITERATIVELY macro 1205
PRL_ERR_CANT_CONVERT_CONFIG 1205
PRL_ERR_CANT_CONVERT_CONFIG macro 1205
PRL_ERR_CANT_CONVERT_HDD_IMAGE_FILE 1206
PRL_ERR_CANT_CONVERT_HDD_IMAGE_FILE macro 1206
PRL_ERR_CANT_CONVERT_OLD_VM_WITH_INSTALLED_TOOLS 1206
PRL_ERR_CANT_CONVERT_OLD_VM_WITH_INSTALLED_TOOLS macro 1206
PRL_ERR_CANT_CONVERT_OTHER_VENDOR_HDD 1206
PRL_ERR_CANT_CONVERT_OTHER_VENDOR_HDD macro 1206
PRL_ERR_CANT_CONVERT_OTHER_VENDOR_VM 1206
PRL_ERR_CANT_CONVERT_OTHER_VENDOR_VM macro 1206
PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_DISKS_PRESENT 1207
PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_DISKS_PRESENT macro 1207
PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_SNAPSHOTS_PRESENT 1207
PRL_ERR_CANT_CONVERT_VM_CONFIG_DUE_UNDO_SNAPSHOTS_PRESENT macro 1207
PRL_ERR_CANT_CREATE_FLOPPY_IMAGE 1207
PRL_ERR_CANT_CREATE_FLOPPY_IMAGE macro 1207
PRL_ERR_CANT_CREATE_HDD_IMAGE 1207
PRL_ERR_CANT_CREATE_HDD_IMAGE macro 1207
PRL_ERR_CANT_CREATE_HDD_IMAGE_NO_SPACE 1208
PRL_ERR_CANT_CREATE_HDD_IMAGE_NO_SPACE macro 1208
PRL_ERR_CANT_CREATE_PARALLEL_PORT_IMAGE 1208
PRL_ERR_CANT_CREATE_PARALLEL_PORT_IMAGE macro 1208
PRL_ERR_CANT_CREATE_SERIAL_PORT_IMAGE 1208
PRL_ERR_CANT_CREATE_SERIAL_PORT_IMAGE macro 1208
PRL_ERR_CANT_DELETE_FILE 1208
PRL_ERR_CANT_DELETE_FILE macro 1208
PRL_ERR_CANT_EDIT_SUSPENDED_VM 1208
PRL_ERR_CANT_EDIT_SUSPENDED_VM macro 1208

PRL_ERR_CANT_GET_FILE_PERMISSIONS 1209

PRL_ERR_CANT_GET_FILE_PERMISSIONS macro 1209

PRL_ERR_CANT_INIT_REAL_CPUS_INFO 1209

PRL_ERR_CANT_INIT_REAL_CPUS_INFO macro 1209

PRL_ERR_CANT_PARSE_DISP_EVENT 1209

PRL_ERR_CANT_PARSE_DISP_EVENT macro 1209

PRL_ERR_CANT_PARSE_VM_CONFIG 1209

PRL_ERR_CANT_PARSE_VM_CONFIG macro 1209

PRL_ERR_CANT_PREPARE_RECONFIG_DATA 1209

PRL_ERR_CANT_PREPARE_RECONFIG_DATA macro 1209

PRL_ERR_CANT_RECONFIG_GUEST_OS 1210

PRL_ERR_CANT_RECONFIG_GUEST_OS macro 1210

PRL_ERR_CANT_RECREATE_HDD 1210

PRL_ERR_CANT_RECREATE_HDD macro 1210

PRL_ERR_CANT_REMOVE_DIR_ENTRY 1210

PRL_ERR_CANT_REMOVE_DIR_ENTRY macro 1210

PRL_ERR_CANT_REMOVE_ENTRY 1210

PRL_ERR_CANT_REMOVE_ENTRY macro 1210

PRL_ERR_CANT_REMOVE_INVALID_VM_AS_NON_ADMIN 1210

PRL_ERR_CANT_REMOVE_INVALID_VM_AS_NON_ADMIN macro 1210

PRL_ERR_CANT_RENAME_DIR_ENTRY 1211

PRL_ERR_CANT_RENAME_DIR_ENTRY macro 1211

PRL_ERR_CANT_RENAME_ENTRY 1211

PRL_ERR_CANT_RENAME_ENTRY macro 1211

PRL_ERR_CANT_REPLACE_FLOPPY_IMAGE 1211

PRL_ERR_CANT_REPLACE_FLOPPY_IMAGE macro 1211

PRL_ERR_CANT_RESIZE_HDD 1211

PRL_ERR_CANT_RESIZE_HDD macro 1211

PRL_ERR_CANT_RESOLVE_HOSTNAME 1212

PRL_ERR_CANT_RESOLVE_HOSTNAME macro 1212

PRL_ERR_CANT_REVERT_VM_SINCE_VTD_DEVICE_ALREADY_USED
1212

PRL_ERR_CANT_REVERT_VM_SINCE_VTD_DEVICE_ALREADY_USED
macro 1212

PRL_ERR_CANT_START_VM_SINCE_NO_VTD_DRIVER 1212

PRL_ERR_CANT_START_VM_SINCE_NO_VTD_DRIVER macro 1212

PRL_ERR_CANT_START_VM_SINCE_VTD_DEVICE_ALREADY_USED
1212

PRL_ERR_CANT_START_VM_SINCE_VTD_DEVICE_ALREADY_USED
macro 1212

PRL_ERR_CANT_SUSPEND_VM_WITH_BOOTCAMP 1213

PRL_ERR_CANT_SUSPEND_VM_WITH_BOOTCAMP macro 1213

PRL_ERR_CANT_TO_CHANGE_PERMISSIONS_ON_REMOTE_LOCATION
1213

PRL_ERR_CANT_TO_CHANGE_PERMISSIONS_ON_REMOTE_LOCATION
macro 1213

PRL_ERR_CANT_TO_START_VM_TEMPLATE 1213

PRL_ERR_CANT_TO_START_VM_TEMPLATE macro 1213

PRL_ERR_CANT_UNPACK_ARCHIVE 1213

PRL_ERR_CANT_UNPACK_ARCHIVE macro 1213

PRL_ERR_CANT_UPDATE_DEVICE_INFO 1214

PRL_ERR_CANT_UPDATE_DEVICE_INFO macro 1214

PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT 1214

PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT macro 1214

PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT32 1214

PRL_ERR_CANTS_CREATE_DISK_IMAGE_ON_FAT32 macro 1214

PRL_ERR_CDD_IMAGE_NOT_SPECIFIED 1214

PRL_ERR_CDD_IMAGE_NOT_SPECIFIED macro 1214

PRL_ERR_CHANGESID_FAILED 1214

PRL_ERR_CHANGESID_FAILED macro 1214

PRL_ERR_CHANGESID_GUEST_TOOLS_NOT_AVAILABLE 1215

PRL_ERR_CHANGESID_GUEST_TOOLS_NOT_AVAILABLE macro 1215

PRL_ERR_CHANGESID_NOT_AVAILABLE 1215

PRL_ERR_CHANGESID_NOT_AVAILABLE macro 1215

PRL_ERR_CHANGESID_NOT_SUPPORTED 1215

PRL_ERR_CHANGESID_NOT_SUPPORTED macro 1215

PRL_ERR_CHANGESID_VM_START_FAILED 1215

PRL_ERR_CHANGESID_VM_START_FAILED macro 1215

PRL_ERR_CI_CANNOT_COPY_IMAGE_NON_STOPPED_VM 1216

PRL_ERR_CI_CANNOT_COPY_IMAGE_NON_STOPPED_VM macro 1216

PRL_ERR_CI_DEVICE_IS_NOT_VIRTUAL 1216

PRL_ERR_CI_DEVICE_IS_NOT_VIRTUAL macro 1216

PRL_ERR_CI_PERMISSIONS_DENIED 1216

PRL_ERR_CI_PERMISSIONS_DENIED macro 1216

PRL_ERR_CLONE_OPERATION_CANCELED 1216

PRL_ERR_CLONE_OPERATION_CANCELED macro 1216

PRL_ERR_CLUSTER_RESOURCE_ERROR 1216

PRL_ERR_CLUSTER_RESOURCE_ERROR macro 1216

PRL_ERR_COM_DEVICE_ALREADY_EXIST 1217

PRL_ERR_COM_DEVICE_ALREADY_EXIST macro 1217

PRL_ERR_COM_OUTPUT_FILE_IS_NOT_SPECIFIED 1217

PRL_ERR_COM_OUTPUT_FILE_IS_NOT_SPECIFIED macro 1217

PRL_ERR_COM_OUTPUT_FILE_NOT_EXIST 1217

PRL_ERR_COM_OUTPUT_FILE_NOT_EXIST macro 1217

PRL_ERR_COMMAND_SUPPORTED_ONLY_AT_SERVER_MODE 1217

PRL_ERR_COMMAND_SUPPORTED_ONLY_AT_SERVER_MODE macro 1217

PRL_ERR_COMMON_SERVER_PREFS_BLOCKED_TO_CHANGE 1218

PRL_ERR_COMMON_SERVER_PREFS_BLOCKED_TO_CHANGE macro 1218

PRL_ERR_COMMON_SERVER_PREFS_WERE_CHANGED 1218

PRL_ERR_COMMON_SERVER_PREFS_WERE_CHANGED macro 1218

PRL_ERR_COMPACT_ALREADY_SWITCHED_ON 1218

PRL_ERR_COMPACT_ALREADY_SWITCHED_ON macro 1218

PRL_ERR_COMPACT_WRONG_VM_STATE 1218

PRL_ERR_COMPACT_WRONG_VM_STATE macro 1218

PRL_ERR_CONCURRENT_COMMAND_EXECUTING 1219

PRL_ERR_CONCURRENT_COMMAND_EXECUTING macro 1219

PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_OBJECT_UUID 1219

PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_OBJECT_UUID macro 1219

PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_USER_UUID 1219

PRL_ERR_CONFIG_BEGIN_EDIT_NOT_FOUND_USER_UUID macro 1219

PRL_ERR_CONFIG_EDIT_COLLISION 1219

PRL_ERR_CONFIG_EDIT_COLLISION macro 1219

PRL_ERR_CONFIG_FILE_NOT_SET 1220

PRL_ERR_CONFIG_FILE_NOT_SET macro 1220

PRL_ERR_CONFIGURE_GENERIC_PCI_TASK_ALREADY_RUN 1220

PRL_ERR_CONFIGURE_GENERIC_PCI_TASK_ALREADY_RUN macro 1220

PRL_ERR_CONFIRMATION_MODE_ALREADY_DISABLED 1220

PRL_ERR_CONFIRMATION_MODE_ALREADY_DISABLED macro 1220

PRL_ERR_CONFIRMATION_MODE_ALREADY_ENABLED 1220

PRL_ERR_CONFIRMATION_MODE_ALREADY_ENABLED macro 1220

PRL_ERR_CONFIRMATION_MODE_UNABLE_CHANGE_BY_NOT_ADMIN 1221

PRL_ERR_CONFIRMATION_MODE_UNABLE_CHANGE_BY_NOT_ADMIN macro 1221

PRL_ERR_CONN_CLIENT_CERTIFICATE_EXPIRED 1221

PRL_ERR_CONN_CLIENT_CERTIFICATE_EXPIRED macro 1221

PRL_ERR_CONN_CLIENT_CERTIFICATE_INVALID 1221

PRL_ERR_CONN_CLIENT_CERTIFICATE_INVALID macro 1221

PRL_ERR_CONN_CLIENT_CERTIFICATE_REVOKED 1221

PRL_ERR_CONN_CLIENT_CERTIFICATE_REVOKED macro 1221

PRL_ERR_CONN_SERVER_CERTIFICATE_EXPIRED 1222

PRL_ERR_CONN_SERVER_CERTIFICATE_EXPIRED macro 1222

PRL_ERR_CONN_SERVER_CERTIFICATE_INVALID 1222

PRL_ERR_CONN_SERVER_CERTIFICATE_INVALID macro 1222

PRL_ERR_CONN_SERVER_CERTIFICATE_REVOKED 1222

PRL_ERR_CONN_SERVER_CERTIFICATE_REVOKED

macro 1222

PRL_ERR_CONN_UNABLE_TO_ESTABLISH_TRUSTED_CHANNEL
1222

PRL_ERR_CONN_UNABLE_TO_ESTABLISH_TRUSTED_CHANNEL
macro 1222

PRL_ERR_CONNECT_TO_MOUNTER 1223

PRL_ERR_CONNECT_TO_MOUNTER macro 1223

PRL_ERR_CONV_HD_CONFLICT 1223

PRL_ERR_CONV_HD_CONFLICT macro 1223

PRL_ERR_CONV_HD_DISK_TOOL_NOT_STARTED 1223

PRL_ERR_CONV_HD_DISK_TOOL_NOT_STARTED macro 1223

PRL_ERR_CONV_HD_EXIT_WITH_ERROR 1223

PRL_ERR_CONV_HD_EXIT_WITH_ERROR macro 1223

PRL_ERR_CONV_HD_NO_ONE_DISK_FOR_CONVERSION 1223

PRL_ERR_CONV_HD_NO_ONE_DISK_FOR_CONVERSION macro 1223

PRL_ERR_CONV_HD_WRONG_VM_STATE 1224

PRL_ERR_CONV_HD_WRONG_VM_STATE macro 1224

PRL_ERR_CONVERT_3RD_PARTY_VM_FAILED 1224

PRL_ERR_CONVERT_3RD_PARTY_VM_FAILED macro 1224

PRL_ERR_CONVERT_3RD_PARTY_VM_NO_SPACE 1224

PRL_ERR_CONVERT_3RD_PARTY_VM_NO_SPACE macro 1224

PRL_ERR_CONVERT_EFI_CONFIG_INVALID 1224

PRL_ERR_CONVERT_EFI_CONFIG_INVALID macro 1224

PRL_ERR_CONVERT_EFI_GUEST_OS_UNSUPPORTED 1225

PRL_ERR_CONVERT_EFI_GUEST_OS_UNSUPPORTED macro 1225

PRL_ERR_CONVERT_GUEST_OS_IS_HIBERNATED 1225

PRL_ERR_CONVERT_GUEST_OS_IS_HIBERNATED macro 1225

PRL_ERR_CONVERT_NO_GUEST_OS_FOUND 1225

PRL_ERR_CONVERT_NO_GUEST_OS_FOUND macro 1225

PRL_ERR_COPY_CT_TMPL_INTERNAL_ERROR 1225

PRL_ERR_COPY_CT_TMPL_INTERNAL_ERROR macro 1225

PRL_ERR_COPY_VM_INFO_FILE 1225

PRL_ERR_COPY_VM_INFO_FILE macro 1225

PRL_ERR_CORE_STATE_CANCELLED 1226

PRL_ERR_CORE_STATE_CANCELLED macro 1226

PRL_ERR_CORE_STATE_CHANGE_VM_CONFIG 1226

PRL_ERR_CORE_STATE_CHANGE_VM_CONFIG macro 1226

PRL_ERR_CORE_STATE_CORRUPT_MEM_FILE 1226

PRL_ERR_CORE_STATE_CORRUPT_MEM_FILE macro 1226

PRL_ERR_CORE_STATE_CORRUPT_SAV_FILE 1226

PRL_ERR_CORE_STATE_CORRUPT_SAV_FILE macro 1226

PRL_ERR_CORE_STATE_ERROR_COMMON 1226

PRL_ERR_CORE_STATE_ERROR_COMMON macro 1226

PRL_ERR_CORE_STATE_INV_SAV_VERSION 1227

PRL_ERR_CORE_STATE_INV_SAV_VERSION macro 1227

PRL_ERR_CORE_STATE_NO_FILE 1227

PRL_ERR_CORE_STATE_NO_FILE macro 1227

PRL_ERR_CORE_STATE_VM_WOULD_RESTART 1227

PRL_ERR_CORE_STATE_VM_WOULD_RESTART macro 1227

PRL_ERR_CORE_STATE_VM_WOULD_STOP 1227

PRL_ERR_CORE_STATE_VM_WOULD_STOP macro 1227

PRL_ERR_COULD_NOT_START_FIREWALL_TOOL 1228

PRL_ERR_COULD_NOT_START_FIREWALL_TOOL macro 1228

PRL_ERR_COULDNT_CREATE_AUTHORIZATION_FILE 1228

PRL_ERR_COULDNT_CREATE_AUTHORIZATION_FILE macro 1228

PRL_ERR_COULDNT_SET_PERMISSIONS_TO_AUTHORIZATION_FILE
1228

PRL_ERR_COULDNT_SET_PERMISSIONS_TO_AUTHORIZATION_FILE
macro 1228

PRL_ERR_COULDNT_TO_CREATE_HDD_LINKED_CLONE 1228

PRL_ERR_COULDNT_TO_CREATE_HDD_LINKED_CLONE macro 1228

PRL_ERR_CPU_RESTART 1229

PRL_ERR_CPU_RESTART macro 1229

PRL_ERR_CPU_SHUTDOWN 1229

PRL_ERR_CPU_SHUTDOWN macro 1229

PRL_ERR_CPUFEATURES_DEFAULT_POOL_ARG 1229

PRL_ERR_CPUFEATURES_DEFAULT_POOL_ARG macro 1229

PRL_ERR_CPUFEATURES_INCOMPATIBLE_NODE 1229

PRL_ERR_CPUFEATURES_INCOMPATIBLE_NODE macro 1229

PRL_ERR_CPUFEATURES_INCORRECT_MASK 1229

PRL_ERR_CPUFEATURES_INCORRECT_MASK macro 1229

PRL_ERR_CPUFEATURES_NO_BINARY 1230

PRL_ERR_CPUFEATURES_NO_BINARY macro 1230

PRL_ERR_CPUFEATURES_NOT_IN_POOLS 1230

PRL_ERR_CPUFEATURES_NOT_IN_POOLS macro 1230

PRL_ERR_CPUFEATURES_POOLS_MANAGMENT 1230

PRL_ERR_CPUFEATURES_POOLS_MANAGMENT macro 1230

PRL_ERR_CPUFEATURES_RUNNING_VM_OR_CT 1230

PRL_ERR_CPUFEATURES_RUNNING_VM_OR_CT macro 1230

PRL_ERR_CREATE_BOOTABLE_ISO 1230

PRL_ERR_CREATE_BOOTABLE_ISO macro 1230

PRL_ERR_CREATE_HARD_DISK_WITH_ZERO_SIZE 1231

PRL_ERR_CREATE_HARD_DISK_WITH_ZERO_SIZE macro 1231

PRL_ERR_CREATE_PRIVELEGED_PROCESS 1231

PRL_ERR_CREATE_PRIVELEGED_PROCESS macro 1231

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT 1231

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT macro 1231

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_PAUSED 1231

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_PAUSED macro 1231

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL 1232

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL macro 1232

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL 1232

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL macro 1232

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS 1232

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS macro 1232

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL 1233

PRL_ERR_CREATE_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL macro 1233

PRL_ERR_CT_IS_RUNNING 1233

PRL_ERR_CT_IS_RUNNING macro 1233

PRL_ERR_CT_MIGRATE_EXTERNAL_PROCESS 1233

PRL_ERR_CT_MIGRATE_EXTERNAL_PROCESS macro 1233

PRL_ERR_CT_MIGRATE_ID_ALREADY_EXIST 1233

PRL_ERR_CT_MIGRATE_ID_ALREADY_EXIST macro 1233

PRL_ERR_CT_MIGRATE_INTERNAL_ERROR 1234

PRL_ERR_CT_MIGRATE_INTERNAL_ERROR macro 1234

PRL_ERR_CT_MIGRATE_TARGET_ALREADY_EXISTS 1234

PRL_ERR_CT_MIGRATE_TARGET_ALREADY_EXISTS macro 1234

PRL_ERR_CT_NOT_FOUND 1234

PRL_ERR_CT_NOT_FOUND macro 1234

PRL_ERR_CVSRC_IO_ERROR 1234

PRL_ERR_CVSRC_IO_ERROR macro 1234

PRL_ERR_CVSRC_NO_CHANNEL 1234

PRL_ERR_CVSRC_NO_CHANNEL macro 1234

PRL_ERR_CVSRC_NO_OPEN_REQUEST 1235

PRL_ERR_CVSRC_NO_OPEN_REQUEST macro 1235

PRL_ERR_DEACTIVATE_NO_INSTALLED_LICENSE 1235

PRL_ERR_DEACTIVATE_NO_INSTALLED_LICENSE macro 1235

PRL_ERR_DEACTIVATE_TRIAL_LICENSE 1235

PRL_ERR_DEACTIVATE_TRIAL_LICENSE macro 1235

PRL_ERR_DEACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE 1235

PRL_ERR_DEACTIVATE_WRONG_TYPE_OF_ACTIVE_LICENSE macro 1235

PRL_ERR_DEACTIVATION_COMMON_SERVER_ERROR 1236

PRL_ERR_DEACTIVATION_COMMON_SERVER_ERROR macro 1236

PRL_ERR_DEACTIVATION_HTTP_REQUEST_FAILED 1236

PRL_ERR_DEACTIVATION_HTTP_REQUEST_FAILED macro

1236

PRL_ERR_DEACTIVATION_OLD_HWID_NOT_FOUND 1236

PRL_ERR_DEACTIVATION_OLD_HWID_NOT_FOUND macro 1236

PRL_ERR_DEACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID
1236

PRL_ERR_DEACTIVATION_SERVER_ACTIVATION_ID_IS_INVALID
macro 1236

PRL_ERR_DEACTIVATION_SERVER_ERROR 1237

PRL_ERR_DEACTIVATION_SERVER_ERROR macro 1237

PRL_ERR_DEACTIVATION_SERVER_KEY_IS_BLACKLISTED

1237

PRL_ERR_DEACTIVATION_SERVER_KEY_IS_BLACKLISTED

macro 1237

PRL_ERR_DEACTIVATION_SERVER_KEY_IS_INVALID 1237

PRL_ERR_DEACTIVATION_SERVER_KEY_IS_INVALID
macro 1237

PRL_ERR_DEACTIVATION_UNABLE_TO_SEND_REQUEST
1237

PRL_ERR_DEACTIVATION_UNABLE_TO_SEND_REQUEST
macro 1237

PRL_ERR_DEACTIVATION_WRONG_SERVER_RESPONSE
1238

PRL_ERR_DEACTIVATION_WRONG_SERVER_RESPONSE
macro 1238

PRL_ERR_DELETE_UNFINISHED_STATE_FAILED 1238

PRL_ERR_DELETE_UNFINISHED_STATE_FAILED macro
1238

PRL_ERR_DELETING_VM_FROM_CATALOGUE 1238

PRL_ERR_DELETING_VM_FROM_CATALOGUE macro 1238

PRL_ERR_DEV_ALREADY_CONNECTED 1238

PRL_ERR_DEV_ALREADY_CONNECTED macro 1238

PRL_ERR_DEV_ALREADY_DISCONNECTED 1239

PRL_ERR_DEV_ALREADY_DISCONNECTED macro 1239

PRL_ERR_DEV_DISABLE_AFTER_LOST 1239

PRL_ERR_DEV_DISABLE_AFTER_LOST macro 1239

PRL_ERR_DEV_FLOPPY_CONNECT_FAILED 1239

PRL_ERR_DEV_FLOPPY_CONNECT_FAILED macro 1239

PRL_ERR_DEV_MAX_CD_EXCEEDED_WITH_DISABLED_CLIENT_SCSII
1239

PRL_ERR_DEV_MAX_CD_EXCEEDED_WITH_DISABLED_CLIENT_SCSII
macro 1239

PRL_ERR_DEV_MAX_CD_HDD_EXCEEDED 1240

PRL_ERR_DEV_MAX_CD_HDD_EXCEEDED macro 1240

PRL_ERR_DEV_MAX_NUMBER_EXCEEDED 1240

PRL_ERR_DEV_MAX_NUMBER_EXCEEDED macro 1240

PRL_ERR_DEV_PARALLEL_PORT_FILE_CONNECT_FAILED

1240

PRL_ERR_DEV_PARALLEL_PORT_FILE_CONNECT_FAILED

macro 1240

PRL_ERR_DEV_PARALLEL_PORT_PHYSICAL_CONNECT_FAILED
1240

PRL_ERR_DEV_PARALLEL_PORT_PHYSICAL_CONNECT_FAILED
macro 1240

PRL_ERR_DEV_PARALLEL_PORT_PRINTER_CONNECT_FAILED
1241

PRL_ERR_DEV_PARALLEL_PORT_PRINTER_CONNECT_FAILED
macro 1241

PRL_ERR_DEV_PARALLEL_PORT_REMOTE_CONNECT_FAILED
1241

PRL_ERR_DEV_PARALLEL_PORT_REMOTE_CONNECT_FAILED
macro 1241

PRL_ERR_DEV_PRINTER_OVERFLOW 1241

PRL_ERR_DEV_PRINTER_OVERFLOW macro 1241

PRL_ERR_DEV_SERIAL_PORT_FILE_CONNECT_FAILED
1241

PRL_ERR_DEV_SERIAL_PORT_FILE_CONNECT_FAILED
macro 1241

PRL_ERR_DEV_SERIAL_PORT_PHYSICAL_CONNECT_FAILED
1242

PRL_ERR_DEV_SERIAL_PORT_PHYSICAL_CONNECT_FAILED
macro 1242

PRL_ERR_DEV_SERIAL_PORT_PIPE_CONNECT_FAILED
1242

PRL_ERR_DEV_SERIAL_PORT_PIPE_CONNECT_FAILED
macro 1242

PRL_ERR_DEV_SERIAL_PORT_REMOTE_CONNECT_FAILED

1242
PRL_ERR_DEV_SERIAL_PORT_REMOTE_CONNECT_FAILED macro 1242
PRL_ERR_DEV_USB_BUSY 1242
PRL_ERR_DEV_USB_BUSY macro 1242
PRL_ERR_DEV_USB_CHANGEPID 1243
PRL_ERR_DEV_USB_CHANGEPID macro 1243
PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED 1243
PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED macro 1243
PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED2 1243
PRL_ERR_DEV_USB_HARD_DEVICE_INSERTED2 macro 1243
PRL_ERR_DEV_USB_INSTALL_DRIVER_FAILED 1243
PRL_ERR_DEV_USB_INSTALL_DRIVER_FAILED macro 1243
PRL_ERR_DEV_USB_NO_FREE_PORTS 1244
PRL_ERR_DEV_USB_NO_FREE_PORTS macro 1244
PRL_ERR_DEV_USB_NOT_CONFIGURED 1244
PRL_ERR_DEV_USB_NOT_CONFIGURED macro 1244
PRL_ERR_DEV_USB_OPEN_MANAGER_FAILED 1244
PRL_ERR_DEV_USB_OPEN_MANAGER_FAILED macro 1244
PRL_ERR_DEV_USB_REUSE 1244
PRL_ERR_DEV_USB_REUSE macro 1244
PRL_ERR_DIRECTORY_DOES_NOT_EXIST 1244
PRL_ERR_DIRECTORY_DOES_NOT_EXIST macro 1244
PRL_ERR_DISK_BLOCK_CREATED 1245
PRL_ERR_DISK_BLOCK_CREATED macro 1245
PRL_ERR_DISK_BLOCK_PARTIALLY_PROCESSED 1245
PRL_ERR_DISK_BLOCK_PARTIALLY_PROCESSED macro 1245
PRL_ERR_DISK_BLOCK_SEARCH_FAILED 1245
PRL_ERR_DISK_BLOCK_SEARCH_FAILED macro 1245
PRL_ERR_DISK_BLOCK_SKIPPED 1245
PRL_ERR_DISK_BLOCK_SKIPPED macro 1245
PRL_ERR_DISK_BOOTCAMP_WRITE_MBR 1246
PRL_ERR_DISK_BOOTCAMP_WRITE_MBR macro 1246
PRL_ERR_DISK_CANT_INITIALIZE_IMAGE 1246
PRL_ERR_DISK_CANT_INITIALIZE_IMAGE macro 1246
PRL_ERR_DISK_COMPRESSED_FILE_EMPTY 1246
PRL_ERR_DISK_COMPRESSED_FILE_EMPTY macro 1246
PRL_ERR_DISK_CREATE_IMAGE_ERROR 1246
PRL_ERR_DISK_CREATE_IMAGE_ERROR macro 1246
PRL_ERR_DISK_DATA_NOT_FOUND 1246
PRL_ERR_DISK_DATA_NOT_FOUND macro 1246
PRL_ERR_DISK_DIR_CREATE_ERROR 1247
PRL_ERR_DISK_DIR_CREATE_ERROR macro 1247
PRL_ERR_DISK_DISK_NOT_OPENED 1247
PRL_ERR_DISK_DISK_NOT_OPENED macro 1247
PRL_ERR_DISK_ENLARGE_FAILED 1247
PRL_ERR_DISK_ENLARGE_FAILED macro 1247
PRL_ERR_DISK_FAT32_SIZE_EXCEEDED 1247
PRL_ERR_DISK_FAT32_SIZE_EXCEEDED macro 1247
PRL_ERR_DISK_FILE_CREATE_ERROR 1248
PRL_ERR_DISK_FILE_CREATE_ERROR macro 1248
PRL_ERR_DISK_FILE_EXISTS 1248
PRL_ERR_DISK_FILE_EXISTS macro 1248
PRL_ERR_DISK_FILE_OPEN_ERROR 1248
PRL_ERR_DISK_FILE_OPEN_ERROR macro 1248
PRL_ERR_DISK_FSYNC_FAILED 1248
PRL_ERR_DISK_FSYNC_FAILED macro 1248
PRL_ERR_DISK_FULLFSYNC_FAILED 1248
PRL_ERR_DISK_FULLFSYNC_FAILED macro 1248
PRL_ERR_DISK_GENERIC_ERROR 1249
PRL_ERR_DISK_GENERIC_ERROR macro 1249
PRL_ERR_DISK_GET_MOUNTPATH_FAILED 1249
PRL_ERR_DISK_GET_MOUNTPATH_FAILED macro 1249
PRL_ERR_DISK_GET_PARAMS_FAILED 1249
PRL_ERR_DISK_GET_PARAMS_FAILED macro 1249
PRL_ERR_DISK_GET_SIZE_FAILED 1249
PRL_ERR_DISK_GET_SIZE_FAILED macro 1249
PRL_ERR_DISK_GPT_MBR_NOT_EQUAL 1250
PRL_ERR_DISK_GPT_MBR_NOT_EQUAL macro 1250
PRL_ERR_DISK_GROUP_INTERSECT 1250
PRL_ERR_DISK_GROUP_INTERSECT macro 1250
PRL_ERR_DISK_IDENTIFY_FAILED 1250
PRL_ERR_DISK_IDENTIFY_FAILED macro 1250
PRL_ERR_DISK_IMAGE_BUSY 1250
PRL_ERR_DISK_IMAGE_BUSY macro 1250
PRL_ERR_DISK_IMAGES_CORRUPTED 1250

PRL_ERR_DISK_IMAGES_CORRUPTED macro 1250
PRL_ERR_DISK_IMPERSONATE_FAILED 1251
PRL_ERR_DISK_IMPERSONATE_FAILED macro 1251
PRL_ERR_DISK_INCORRECTLY_CLOSED 1251
PRL_ERR_DISK_INCORRECTLY_CLOSED macro 1251
PRL_ERR_DISK_INSUFFICIENT_SPACE 1251
PRL_ERR_DISK_INSUFFICIENT_SPACE macro 1251
PRL_ERR_DISK_INTERNAL_CLASS_ERROR 1251
PRL_ERR_DISK_INTERNAL_CLASS_ERROR macro 1251
PRL_ERR_DISK_INVALID_BLOCKSIZE 1251
PRL_ERR_DISK_INVALID_BLOCKSIZE macro 1251
PRL_ERR_DISK_INVALID_FORMAT 1252
PRL_ERR_DISK_INVALID_FORMAT macro 1252
PRL_ERR_DISK_INVALID_PARAMETERS 1252
PRL_ERR_DISK_INVALID_PARAMETERS macro 1252
PRL_ERR_DISK_MEMORY_ERROR 1252
PRL_ERR_DISK_MEMORY_ERROR macro 1252
PRL_ERR_DISK_MOUNT_FAILED 1252
PRL_ERR_DISK_MOUNT_FAILED macro 1252
PRL_ERR_DISK_MOUNTED_OVERLAP 1252
PRL_ERR_DISK_MOUNTED_OVERLAP macro 1252
PRL_ERR_DISK_NOT_IMPLEMENTED 1253
PRL_ERR_DISK_NOT_IMPLEMENTED macro 1253
PRL_ERR_DISK_NOT_PERMITTED 1253
PRL_ERR_DISK_NOT_PERMITTED macro 1253
PRL_ERR_DISK_NOT_VALID_OFFSET 1253
PRL_ERR_DISK_NOT_VALID_OFFSET macro 1253
PRL_ERR_DISK_NULL_PART_SIZE 1253
PRL_ERR_DISK_NULL_PART_SIZE macro 1253
PRL_ERR_DISK_OFFSETS_FIXED 1253
PRL_ERR_DISK_OFFSETS_FIXED macro 1253
PRL_ERR_DISK_OPERATION_ABORTED 1254
PRL_ERR_DISK_OPERATION_ABORTED macro 1254
PRL_ERR_DISK_OPERATION_IN_PROGRESS 1254
PRL_ERR_DISK_OPERATION_IN_PROGRESS macro 1254
PRL_ERR_DISK_OPERATION_NOT_ALLOWED 1254
PRL_ERR_DISK_OPERATION_NOT_ALLOWED macro 1254
PRL_ERR_DISK_PARTITION_INVALID_NAME 1254
PRL_ERR_DISK_PARTITION_INVALID_NAME macro 1254
PRL_ERR_DISK_PARTITION_NOT_FOUND 1254
PRL_ERR_DISK_PARTITION_NOT_FOUND macro 1254
PRL_ERR_DISK_PARTITIONS_TABLE_CYCLE 1255
PRL_ERR_DISK_PARTITIONS_TABLE_CYCLE macro 1255
PRL_ERR_DISK_POINTERS_MIXED_UP 1255
PRL_ERR_DISK_POINTERS_MIXED_UP macro 1255
PRL_ERR_DISK_POSSIBLE_OVERRUN 1255
PRL_ERR_DISK_POSSIBLE_OVERRUN macro 1255
PRL_ERR_DISK_READ_FAILED 1255
PRL_ERR_DISK_READ_FAILED macro 1255
PRL_ERR_DISK_READ_OUT_DISK 1255
PRL_ERR_DISK_READ_OUT_DISK macro 1255
PRL_ERR_DISK_RENAME_ERROR 1256
PRL_ERR_DISK_RENAME_ERROR macro 1256
PRL_ERR_DISK_RESERVED_WORD 1256
PRL_ERR_DISK_RESERVED_WORD macro 1256
PRL_ERR_DISK_RESIZE_FAILED 1256
PRL_ERR_DISK_RESIZE_FAILED macro 1256
PRL_ERR_DISK_RESIZE_SIZE_TOO_LOW 1256
PRL_ERR_DISK_RESIZE_SIZE_TOO_LOW macro 1256
PRL_ERR_DISK_RESIZE_WITH_SNAPSHOTS_NOT_ALLOWED
1257
PRL_ERR_DISK_RESIZE_WITH_SNAPSHOTS_NOT_ALLOWED
macro 1257
PRL_ERR_DISK_RESIZER_NOT_FOUND 1257
PRL_ERR_DISK_RESIZER_NOT_FOUND macro 1257
PRL_ERR_DISK_SET_CACHING_FAILED 1257
PRL_ERR_DISK_SET_CACHING_FAILED macro 1257
PRL_ERR_DISK_SET_FILE_SIZE_FAILED 1257
PRL_ERR_DISK_SET_FILE_SIZE_FAILED macro 1257
PRL_ERR_DISK_SHARED_BLOCK 1258
PRL_ERR_DISK_SHARED_BLOCK macro 1258
PRL_ERR_DISK_SHARING_VIOLATION 1258
PRL_ERR_DISK_SHARING_VIOLATION macro 1258
PRL_ERR_DISK_SMALL_IMAGE_SIZE 1258
PRL_ERR_DISK_SMALL_IMAGE_SIZE macro 1258
PRL_ERR_DISK_SNAPSHOTS_CORRUPTED 1258
PRL_ERR_DISK_SNAPSHOTS_CORRUPTED macro 1258
PRL_ERR_DISK_STATES_ERROR 1258

PRL_ERR_DISK_STATES_ERROR macro 1258
PRL_ERR_DISK_STORAGE_CORRUPTED 1259
PRL_ERR_DISK_STORAGE_CORRUPTED macro 1259
PRL_ERR_DISK_TOOL_NOT_FOUND 1259
PRL_ERR_DISK_TOOL_NOT_FOUND macro 1259
PRL_ERR_DISK_TOOL_PROCESS_ERROR 1259
PRL_ERR_DISK_TOOL_PROCESS_ERROR macro 1259
PRL_ERR_DISK_TRUNCATE_FAILED 1259
PRL_ERR_DISK_TRUNCATE_FAILED macro 1259
PRL_ERR_DISK_UNALIGNED 1260
PRL_ERR_DISK_UNALIGNED macro 1260
PRL_ERR_DISK_UNCOMMITED_OPERATION 1260
PRL_ERR_DISK_UNCOMMITED_OPERATION macro 1260
PRL_ERR_DISK_UNMOUNT_FAILED 1260
PRL_ERR_DISK_UNMOUNT_FAILED macro 1260
PRL_ERR_DISK_USER_INTERRUPTED 1260
PRL_ERR_DISK_USER_INTERRUPTED macro 1260
PRL_ERR_DISK_WRITE_FAILED 1260
PRL_ERR_DISK_WRITE_FAILED macro 1260
PRL_ERR_DISK_WRITE_OUT_DISK 1261
PRL_ERR_DISK_WRITE_OUT_DISK macro 1261
PRL_ERR_DISK_WRITE_REAL_FAILED 1261
PRL_ERR_DISK_WRITE_REAL_FAILED macro 1261
PRL_ERR_DISK_XML_DIFFERS_FROM_REAL 1261
PRL_ERR_DISK_XML_DIFFERS_FROM_REAL macro 1261
PRL_ERR_DISK_XML_INVALID 1261
PRL_ERR_DISK_XML_INVALID macro 1261
PRL_ERR_DISK_XML_INVALID_VERSION 1261
PRL_ERR_DISK_XML_INVALID_VERSION macro 1261
PRL_ERR_DISK_XML_LARGE_FILE 1262
PRL_ERR_DISK_XML_LARGE_FILE macro 1262
PRL_ERR_DISK_XML_LOCKED 1262
PRL_ERR_DISK_XML_LOCKED macro 1262
PRL_ERR_DISK_XML_MISSING 1262
PRL_ERR_DISK_XML_MISSING macro 1262
PRL_ERR_DISK_XML_OPEN_FAILED 1262
PRL_ERR_DISK_XML_OPEN_FAILED macro 1262
PRL_ERR_DISK_XML_PARTITION_NOT_FOUND 1262
PRL_ERR_DISK_XML_PARTITION_NOT_FOUND macro 1262
PRL_ERR_DISK_XML_SAVE_ERROR 1263
PRL_ERR_DISK_XML_SAVE_ERROR macro 1263
PRL_ERR_DISK_XML_SAVE_REMOVE_ERROR 1263
PRL_ERR_DISK_XML_SAVE_REMOVE_ERROR macro 1263
PRL_ERR_DISK_XML_SAVE_RENAME_ERROR 1263
PRL_ERR_DISK_XML_SAVE_RENAME_ERROR macro 1263
PRL_ERR_DISP_AUTO_COMPRESS_PERIOD_OUT_OF_RANGE
1263
PRL_ERR_DISP_AUTO_COMPRESS_PERIOD_OUT_OF_RANGE
macro 1263
PRL_ERR_DISP_CANNOT_COMPACT_VM_HARD_DISK
1264
PRL_ERR_DISP_CANNOT_COMPACT_VM_HARD_DISK
macro 1264
PRL_ERR_DISP_CONFIG_ALREADY_EXISTS 1264
PRL_ERR_DISP_CONFIG_ALREADY_EXISTS macro 1264
PRL_ERR_DISP_CONFIG_FILE_NOT_SET 1264
PRL_ERR_DISP_CONFIG_FILE_NOT_SET macro 1264
PRL_ERR_DISP_CONFIG_WRITE_ERR 1264
PRL_ERR_DISP_CONFIG_WRITE_ERR macro 1264
PRL_ERR_DISP_LOGON_ACTIONS_REACHED_UP_LIMIT
1265
PRL_ERR_DISP_LOGON_ACTIONS_REACHED_UP_LIMIT
macro 1265
PRL_ERR_DISP_SHUTDOWN_IN_PROCESS 1265
PRL_ERR_DISP_SHUTDOWN_IN_PROCESS macro 1265
PRL_ERR_DISP_START_VM_FOR_COMPACT_FAILED 1265
PRL_ERR_DISP_START_VM_FOR_COMPACT_FAILED
macro 1265
PRL_ERR_DISP_TIME_MACHINE_IS_RUNNING 1265
PRL_ERR_DISP_TIME_MACHINE_IS_RUNNING macro 1265
PRL_ERR_DISP_VM_COMMAND_CANT_BE_EXECUTED
1266
PRL_ERR_DISP_VM_COMMAND_CANT_BE_EXECUTED
macro 1266
PRL_ERR_DISP_VM_IS_NOT_STARTED 1266
PRL_ERR_DISP_VM_IS_NOT_STARTED macro 1266
PRL_ERR_DISP_VM_IS_NOT_STOPPED 1266
PRL_ERR_DISP_VM_IS_NOT_STOPPED macro 1266
PRL_ERR_DISP2DISP_SESSION_ALREADY_AUTHORIZED
1266
PRL_ERR_DISP2DISP_SESSION_ALREADY_AUTHORIZED

macro 1266
PRL_ERR_DISP2DISP_WRONG_USER_SESSION_UUID 1267
PRL_ERR_DISP2DISP_WRONG_USER_SESSION_UUID macro 1267
PRL_ERR_DISPATCHER_SERVICE_MODE 1267
PRL_ERR_DISPATCHER_SERVICE_MODE macro 1267
PRL_ERR_DISPLAY_ENCODINGS_NOT_SUPPORTED 1267
PRL_ERR_DISPLAY_ENCODINGS_NOT_SUPPORTED macro 1267
PRL_ERR_DISPLAY_FORMAT_NOT_SUPPORTED 1267
PRL_ERR_DISPLAY_FORMAT_NOT_SUPPORTED macro 1267
PRL_ERR_DOUBLE_INIT 1267
PRL_ERR_DOUBLE_INIT macro 1267
PRL_ERR_DROP_SUSPEND_FOR_HDD_QUEST 1268
PRL_ERR_DROP_SUSPEND_FOR_HDD_QUEST macro 1268
PRL_ERR_ENCRYPTION_COMMIT_PROHIBITED 1268
PRL_ERR_ENCRYPTION_COMMIT_PROHIBITED macro 1268
PRL_ERR_ENCRYPTION_COMMIT_REJECTED 1268
PRL_ERR_ENCRYPTION_COMMIT_REJECTED macro 1268
PRL_ERR_ENTRY_ALREADY_EXISTS 1268
PRL_ERR_ENTRY_ALREADY_EXISTS macro 1268
PRL_ERR_ENTRY_DIR_ALREADY_EXISTS 1269
PRL_ERR_ENTRY_DIR_ALREADY_EXISTS macro 1269
PRL_ERR_ENTRY_DIR_DOES_NOT_EXIST 1269
PRL_ERR_ENTRY_DIR_DOES_NOT_EXIST macro 1269
PRL_ERR_ENTRY_DOES_NOT_EXIST 1269
PRL_ERR_ENTRY_DOES_NOT_EXIST macro 1269
PRL_ERR_EXCEED_LIMIT_MAX_RUNNING_VMS 1269
PRL_ERR_EXCEED_LIMIT_MAX_RUNNING_VMS macro 1269
PRL_ERR_EXCEED_MEMORY_LIMIT 1269
PRL_ERR_EXCEED_MEMORY_LIMIT macro 1269
PRL_ERR_EXT_DISK_ALREADY_EXISTS 1270
PRL_ERR_EXT_DISK_ALREADY_EXISTS macro 1270
PRL_ERR_EXT_DISK_CANT_RENAME 1270
PRL_ERR_EXT_DISK_CANT_RENAME macro 1270
PRL_ERR_FAILED_TO_AUTH_ON_BACKUP_SERVER 1270
PRL_ERR_FAILED_TO_AUTH_ON_BACKUP_SERVER macro 1270
PRL_ERR_FAILED_TO_CONNECT_TO_BACKUP_SERVER 1270
PRL_ERR_FAILED_TO_CONNECT_TO_BACKUP_SERVER macro 1270
PRL_ERR_FAILED_TO_START_VNC_SERVER 1270
PRL_ERR_FAILED_TO_START_VNC_SERVER macro 1270
PRL_ERR_FAILED_TO_STOP_VNC_SERVER 1271
PRL_ERR_FAILED_TO_STOP_VNC_SERVER macro 1271
PRL_ERR_FAILURE 1271
PRL_ERR_FAILURE macro 1271
PRL_ERR_FAILURE_ON_VM_DESTROYED 1271
PRL_ERR_FAILURE_ON_VM_DESTROYED macro 1271
PRL_ERR_FDD_IMAGE_CLONE_TO_SELF 1271
PRL_ERR_FDD_IMAGE_CLONE_TO_SELF macro 1271
PRL_ERR_FDD_IMAGE_COPY 1272
PRL_ERR_FDD_IMAGE_COPY macro 1272
PRL_ERR_FDD_IMAGE_NOT_SPECIFIED 1272
PRL_ERR_FDD_IMAGE_NOT_SPECIFIED macro 1272
PRL_ERR_FILE_DISK_SPACE_ERROR 1272
PRL_ERR_FILE_DISK_SPACE_ERROR macro 1272
PRL_ERR_FILE_NOT_EXIST 1272
PRL_ERR_FILE_NOT_EXIST macro 1272
PRL_ERR_FILE_NOT_FOUND 1272
PRL_ERR_FILE_NOT_FOUND macro 1272
PRL_ERR_FILE_OR_DIR_ALREADY_EXISTS 1273
PRL_ERR_FILE_OR_DIR_ALREADY_EXISTS macro 1273
PRL_ERR_FILE_READ_ERROR 1273
PRL_ERR_FILE_READ_ERROR macro 1273
PRL_ERR_FILE_TRANSFER_CANT_CREATE_DST_FILE 1273
PRL_ERR_FILE_TRANSFER_CANT_CREATE_DST_FILE macro 1273
PRL_ERR_FILE_TRANSFER_CANT_LOCATE_SRC_FILE 1273
PRL_ERR_FILE_TRANSFER_CANT_LOCATE_SRC_FILE macro 1273
PRL_ERR_FILE_TRANSFER_CANT_READ_SRC_FILE 1274
PRL_ERR_FILE_TRANSFER_CANT_READ_SRC_FILE macro 1274
PRL_ERR_FILE_TRANSFER_CANT_WRITE_DATA_TO_DST_FILE 1274
PRL_ERR_FILE_TRANSFER_CANT_WRITE_DATA_TO_DST_

FILE
macro 1274

PRL_ERR_FILE_TRANSFER_CLIENT_NOT_CONNECTED
1274

PRL_ERR_FILE_TRANSFER_CLIENT_NOT_CONNECTED
macro 1274

PRL_ERR_FILE_TRANSFER_DST_FILE_ALREADY_EXIST
1274

PRL_ERR_FILE_TRANSFER_DST_FILE_ALREADY_EXIST
macro 1274

PRL_ERR_FILE_TRANSFER_INVALID_ARGUMENTS 1275

PRL_ERR_FILE_TRANSFER_INVALID_ARGUMENTS macro
1275

PRL_ERR_FILE_TRANSFER_INVALID_CREDENTIALS 1275

PRL_ERR_FILE_TRANSFER_INVALID_CREDENTIALS macro
1275

PRL_ERR_FILE_TRANSFER_OPERATION_NOT_SUPPORTE
D
1275

PRL_ERR_FILE_TRANSFER_OPERATION_NOT_SUPPORTE
D
macro 1275

PRL_ERR_FILE_TRANSFER_UPLOAD_CANCELED_BY_USE
R
1275

PRL_ERR_FILE_TRANSFER_UPLOAD_CANCELED_BY_USE
R
macro 1275

PRL_ERR_FILE_WRITE_ERROR 1276

PRL_ERR_FILE_WRITE_ERROR macro 1276

PRL_ERR_FILECOPY_CANT_CREATE_DIR 1276

PRL_ERR_FILECOPY_CANT_CREATE_DIR macro 1276

PRL_ERR_FILECOPY_CANT_OPEN_FILE 1276

PRL_ERR_FILECOPY_CANT_OPEN_FILE macro 1276

PRL_ERR_FILECOPY_CANT_WRITE 1276

PRL_ERR_FILECOPY_CANT_WRITE macro 1276

PRL_ERR_FILECOPY_DIR_EXIST 1276

PRL_ERR_FILECOPY_DIR_EXIST macro 1276

PRL_ERR_FILECOPY_FILE_EXIST 1277

PRL_ERR_FILECOPY_FILE_EXIST macro 1277

PRL_ERR_FILECOPY_INTERNAL 1277

PRL_ERR_FILECOPY_INTERNAL macro 1277

PRL_ERR_FILECOPY_PROTOCOL 1277

PRL_ERR_FILECOPY_PROTOCOL macro 1277

PRL_ERR_FIREWALL_TOOL_EXECUTED_WITH_ERROR
1277

PRL_ERR_FIREWALL_TOOL_EXECUTED_WITH_ERROR
macro 1277

PRL_ERR_FIXME 1277

PRL_ERR_FIXME macro 1277

PRL_ERR_FLOPPY_DRIVE_INVALID 1278

PRL_ERR_FLOPPY_DRIVE_INVALID macro 1278

PRL_ERR_FLOPPY_IMAGE_ALREADY_EXIST 1278

PRL_ERR_FLOPPY_IMAGE_ALREADY_EXIST macro 1278

PRL_ERR_FLOPPY_IMAGE_NOT_EXIST 1278

PRL_ERR_FLOPPY_IMAGE_NOT_EXIST macro 1278

PRL_ERR_FORCE_REG_ON_SHARED_STORAGE_IS_NEED
ED
1278

PRL_ERR_FORCE_REG_ON_SHARED_STORAGE_IS_NEED
ED
macro 1278

PRL_ERR_FREE_DISC_SPACE_FOR_CLONE 1279

PRL_ERR_FREE_DISC_SPACE_FOR_CLONE macro 1279

PRL_ERR_FREE_DISK_SPACE_FOR_CREATE_SNAPSHOT
1279

PRL_ERR_FREE_DISK_SPACE_FOR_CREATE_SNAPSHOT
macro 1279

PRL_ERR_FREE_DISK_SPACE_FOR_REVERT_TO_SNAPSH
OT
1279

PRL_ERR_FREE_DISK_SPACE_FOR_REVERT_TO_SNAPSH
OT
macro 1279

PRL_ERR_GET_DISK_FREE_SPACE_FAILED 1279

PRL_ERR_GET_DISK_FREE_SPACE_FAILED macro 1279

PRL_ERR_GET_LICENSE_INVALID_ARG 1280

PRL_ERR_GET_LICENSE_INVALID_ARG macro 1280

PRL_ERR_GET_MON_STATE_INVALID_ARG 1280

PRL_ERR_GET_MON_STATE_INVALID_ARG macro 1280

PRL_ERR_GET_MON_STATE_VM_NOT_CONFIGURED 1280

PRL_ERR_GET_MON_STATE_VM_NOT_CONFIGURED
macro 1280

PRL_ERR_GET_MON_STATE_VM_NOT_CREATED 1280

PRL_ERR_GET_MON_STATE_VM_NOT_CREATED macro
1280

PRL_ERR_GET_NET_SERVICE_STATUS_FAILED 1280

PRL_ERR_GET_NET_SERVICE_STATUS_FAILED macro
1280

PRL_ERR_GET_RESOLUTION_TOOL_INVALID_ARG 1281
PRL_ERR_GET_RESOLUTION_TOOL_INVALID_ARG macro 1281
PRL_ERR_GET_RESOLUTION_TOOL_VMNOTCREATED 1281
PRL_ERR_GET_RESOLUTION_TOOL_VMNOTCREATED macro 1281
PRL_ERR_GET_USER_HOME_DIR 1281
PRL_ERR_GET_USER_HOME_DIR macro 1281
PRL_ERR_GUEST_MAC_INVALID_VERSION 1281
PRL_ERR_GUEST_MAC_INVALID_VERSION macro 1281
PRL_ERR_GUEST_MAC_NOT_ENOUGH_MEMORY 1281
PRL_ERR_GUEST_MAC_NOT_ENOUGH_MEMORY macro 1281
PRL_ERR_GUEST_MAC_NOT_MACSERVER_HOST 1282
PRL_ERR_GUEST_MAC_NOT_MACSERVER_HOST macro 1282
PRL_ERR_GUEST_PROGRAM_EXECUTION_FAILED 1282
PRL_ERR_GUEST_PROGRAM_EXECUTION_FAILED macro 1282
PRL_ERR_GUEST_TOOLS_NOT_INSTALLED 1282
PRL_ERR_GUEST_TOOLS_NOT_INSTALLED macro 1282
PRL_ERR_HANDSHAKE_FAILED 1282
PRL_ERR_HANDSHAKE_FAILED macro 1282
PRL_ERR_HARD_DISK_IMAGE_CORRUPTED 1283
PRL_ERR_HARD_DISK_IMAGE_CORRUPTED macro 1283
PRL_ERR_HARD_DISK_NOT_VIRTUAL_OR_DISABLED 1283
PRL_ERR_HARD_DISK_NOT_VIRTUAL_OR_DISABLED macro 1283
PRL_ERR_HDD_IMAGE_CLONE_TO_SELF 1283
PRL_ERR_HDD_IMAGE_CLONE_TO_SELF macro 1283
PRL_ERR_HDD_IMAGE_COPY 1283
PRL_ERR_HDD_IMAGE_COPY macro 1283
PRL_ERR_HDD_IMAGE_IS_ALREADY_EXIST 1283
PRL_ERR_HDD_IMAGE_IS_ALREADY_EXIST macro 1283
PRL_ERR_HDD_IMAGE_NOT_EXIST 1284
PRL_ERR_HDD_IMAGE_NOT_EXIST macro 1284
PRL_ERR_HDD_IMAGE_NOT_SPECIFIED 1284
PRL_ERR_HDD_IMAGE_NOT_SPECIFIED macro 1284
PRL_ERR_HOST_AMD_GUEST_MAC 1284
PRL_ERR_HOST_AMD_GUEST_MAC macro 1284
PRL_ERR_HTTP_AUTH_REQUIRED 1284
PRL_ERR_HTTP_AUTH_REQUIRED macro 1284
PRL_ERR_HTTP_CONNECTION_REFUSED 1285
PRL_ERR_HTTP_CONNECTION_REFUSED macro 1285
PRL_ERR_HTTP_HOST_NOT_FOUND 1285
PRL_ERR_HTTP_HOST_NOT_FOUND macro 1285
PRL_ERR_HTTP_INVALID_RESPONSE_HEADER 1285
PRL_ERR_HTTP_INVALID_RESPONSE_HEADER macro 1285
PRL_ERR_HTTP_PROBLEM_REPORT_SEND_FAILURE 1285
PRL_ERR_HTTP_PROBLEM_REPORT_SEND_FAILURE macro 1285
PRL_ERR_HTTP_PROXY_AUTH_REQUIRED 1286
PRL_ERR_HTTP_PROXY_AUTH_REQUIRED macro 1286
PRL_ERR_HTTP_REQUEST_FAILED 1286
PRL_ERR_HTTP_REQUEST_FAILED macro 1286
PRL_ERR_HTTP_UNEXPECTED_CLOSE 1286
PRL_ERR_HTTP_UNEXPECTED_CLOSE macro 1286
PRL_ERR_HTTP_WRONG_CONTENT_LENGTH 1286
PRL_ERR_HTTP_WRONG_CONTENT_LENGTH macro 1286
PRL_ERR_HVT_DISABLED 1286
PRL_ERR_HVT_DISABLED macro 1286
PRL_ERR_HVT_DISABLED_WARNING 1287
PRL_ERR_HVT_DISABLED_WARNING macro 1287
PRL_ERR_HVT_NOT_AVAILABLE_WARNING 1287
PRL_ERR_HVT_NOT_AVAILABLE_WARNING macro 1287
PRL_ERR_HVT_NOT_PRESENT 1287
PRL_ERR_HVT_NOT_PRESENT macro 1287
PRL_ERR_HVT_NOT_PRESENT_WARNING 1287
PRL_ERR_HVT_NOT_PRESENT_WARNING macro 1287
PRL_ERR_HVT_TURNED_OFF_IN_CONFIG 1287
PRL_ERR_HVT_TURNED_OFF_IN_CONFIG macro 1287
PRL_ERR_HYP_ALLOC_VMS_MEMORY 1288
PRL_ERR_HYP_ALLOC_VMS_MEMORY macro 1288
PRL_ERR_IMAGE_NEED_TO_CONVERT 1288
PRL_ERR_IMAGE_NEED_TO_CONVERT macro 1288
PRL_ERR_IMPERSONATE_FAILED 1288
PRL_ERR_IMPERSONATE_FAILED macro 1288
PRL_ERR_IMPORT_BOOTCAMP_VM_FAILED 1288
PRL_ERR_IMPORT_BOOTCAMP_VM_FAILED macro 1288
PRL_ERR_IMPORT_BOOTCAMP_VM_NO_SPACE 1288
PRL_ERR_IMPORT_BOOTCAMP_VM_NO_SPACE macro

1288
PRL_ERR_INCONSISTENCY_VM_CONFIG 1289
PRL_ERR_INCONSISTENCY_VM_CONFIG macro 1289
PRL_ERR_INCORRECT_CDROM_PATH 1289
PRL_ERR_INCORRECT_CDROM_PATH macro 1289
PRL_ERR_INCORRECT_FDD_PATH 1289
PRL_ERR_INCORRECT_FDD_PATH macro 1289
PRL_ERR_INCORRECT_PATH 1289
PRL_ERR_INCORRECT_PATH macro 1289
PRL_ERR_INSERTING_VM_TO_CATALOGUE 1289
PRL_ERR_INSERTING_VM_TO_CATALOGUE macro 1289
PRL_ERR_INSTALLATION_PROBLEM 1290
PRL_ERR_INSTALLATION_PROBLEM macro 1290
PRL_ERR_INVALID_ACCESS_TOKEN_RECEIVED 1290
PRL_ERR_INVALID_ACCESS_TOKEN_RECEIVED macro 1290
PRL_ERR_INVALID_ACTION_REQUESTED 1290
PRL_ERR_INVALID_ACTION_REQUESTED macro 1290
PRL_ERR_INVALID_ARG 1290
PRL_ERR_INVALID_ARG macro 1290
PRL_ERR_INVALID_CONFIGURATION 1290
PRL_ERR_INVALID_CONFIGURATION macro 1290
PRL_ERR_INVALID_CREATE_FLOPPY_IMAGE_PARAMETERS 1291
PRL_ERR_INVALID_CREATE_FLOPPY_IMAGE_PARAMETERS macro 1291
PRL_ERR_INVALID_HANDLE 1291
PRL_ERR_INVALID_HANDLE macro 1291
PRL_ERR_INVALID_HDD_GEOMETRY 1291
PRL_ERR_INVALID_HDD_GEOMETRY macro 1291
PRL_ERR_INVALID_KEXT_REBOOT_REQUIRED 1291
PRL_ERR_INVALID_KEXT_REBOOT_REQUIRED macro 1291
PRL_ERR_INVALID_MEMORY_GUARANTEE 1292
PRL_ERR_INVALID_MEMORY_GUARANTEE macro 1292
PRL_ERR_INVALID_MEMORY_QUOTA 1292
PRL_ERR_INVALID_MEMORY_QUOTA macro 1292
PRL_ERR_INVALID_MEMORY_SIZE 1292
PRL_ERR_INVALID_MEMORY_SIZE macro 1292
PRL_ERR_INVALID_OS_TYPE 1292
PRL_ERR_INVALID_OS_TYPE macro 1292
PRL_ERR_INVALID_PARALLELS_DISK 1292
PRL_ERR_INVALID_PARALLELS_DISK macro 1292
PRL_ERR_INVALID_PARAM 1293
PRL_ERR_INVALID_PARAM macro 1293
PRL_ERR_INVALID_TOTAL_LIMIT 1293
PRL_ERR_INVALID_TOTAL_LIMIT macro 1293
PRL_ERR_IO_AUTHENTICATION_FAILED 1293
PRL_ERR_IO_AUTHENTICATION_FAILED macro 1293
PRL_ERR_IO_CONNECTION_TIMEOUT 1293
PRL_ERR_IO_CONNECTION_TIMEOUT macro 1293
PRL_ERR_IO_DISABLED 1294
PRL_ERR_IO_DISABLED macro 1294
PRL_ERR_IO_INVALID_POINTER_ACCESS 1294
PRL_ERR_IO_INVALID_POINTER_ACCESS macro 1294
PRL_ERR_IO_NO_CONNECTION 1294
PRL_ERR_IO_NO_CONNECTION macro 1294
PRL_ERR_IO_SEND_QUEUE_IS_FULL 1294
PRL_ERR_IO_SEND_QUEUE_IS_FULL macro 1294
PRL_ERR_IO_STOPPED 1294
PRL_ERR_IO_STOPPED macro 1294
PRL_ERR_IO_UNKNOWN_VM_ID 1295
PRL_ERR_IO_UNKNOWN_VM_ID macro 1295
PRL_ERR_IOSERVICE_COMPRESS 1295
PRL_ERR_IOSERVICE_COMPRESS macro 1295
PRL_ERR_IPC_ATTACH_FAILED 1295
PRL_ERR_IPC_ATTACH_FAILED macro 1295
PRL_ERR_IPC_CREATE_MAP_FAILED 1295
PRL_ERR_IPC_CREATE_MAP_FAILED macro 1295
PRL_ERR_IPC_FTOK_FAILED 1295
PRL_ERR_IPC_FTOK_FAILED macro 1295
PRL_ERR_IPFO_INVALID_MODE 1296
PRL_ERR_IPFO_INVALID_MODE macro 1296
PRL_ERR_IPFO_RECEIVE_FAILED 1296
PRL_ERR_IPFO_RECEIVE_FAILED macro 1296
PRL_ERR_IPFO_SEND_FAILED 1296
PRL_ERR_IPFO_SEND_FAILED macro 1296
PRL_ERR_IPFO_SOCKET_ACCEPT_FAILED 1296
PRL_ERR_IPFO_SOCKET_ACCEPT_FAILED macro 1296

PRL_ERR_IPFO_SOCKET_BIND_FAILED 1296
PRL_ERR_IPFO_SOCKET_BIND_FAILED macro 1296
PRL_ERR_IPFO_SOCKET_CONNECT_FAILED 1297
PRL_ERR_IPFO_SOCKET_CONNECT_FAILED macro 1297
PRL_ERR_IPFO_SOCKET_CREATE_FAILED 1297
PRL_ERR_IPFO_SOCKET_CREATE_FAILED macro 1297
PRL_ERR_IPFO_SOCKET_LISTEN_FAILED 1297
PRL_ERR_IPFO_SOCKET_LISTEN_FAILED macro 1297
PRL_ERR_IPFO_SOCKET_NOT_OPENED 1297
PRL_ERR_IPFO_SOCKET_NOT_OPENED macro 1297
PRL_ERR_IPHONE_PROXY_ALREADY_STARTED 1297
PRL_ERR_IPHONE_PROXY_ALREADY_STARTED macro 1297
PRL_ERR_IPHONE_PROXY_CANNOT_START 1298
PRL_ERR_IPHONE_PROXY_CANNOT_START macro 1298
PRL_ERR_IPHONE_PROXY_CANNOT_STOP 1298
PRL_ERR_IPHONE_PROXY_CANNOT_STOP macro 1298
PRL_ERR_ISCSI_STORAGE_ALREADY_REGISTERED 1298
PRL_ERR_ISCSI_STORAGE_ALREADY_REGISTERED macro 1298
PRL_ERR_ISCSI_STORAGE_CANNOT_CREATE_MOUNT_POINT 1298
PRL_ERR_ISCSI_STORAGE_CANNOT_CREATE_MOUNT_POINT macro 1298
PRL_ERR_ISCSI_STORAGE_CREATE 1299
PRL_ERR_ISCSI_STORAGE_CREATE macro 1299
PRL_ERR_ISCSI_STORAGE_EXTEND 1299
PRL_ERR_ISCSI_STORAGE_EXTEND macro 1299
PRL_ERR_ISCSI_STORAGE_GET_STATE 1299
PRL_ERR_ISCSI_STORAGE_GET_STATE macro 1299
PRL_ERR_ISCSI_STORAGE_INVALID_FSTYPE 1299
PRL_ERR_ISCSI_STORAGE_INVALID_FSTYPE macro 1299
PRL_ERR_ISCSI_STORAGE_LIBRARY 1299
PRL_ERR_ISCSI_STORAGE_LIBRARY macro 1299
PRL_ERR_ISCSI_STORAGE_MOUNT 1300
PRL_ERR_ISCSI_STORAGE_MOUNT macro 1300
PRL_ERR_ISCSI_STORAGE_MOUNT_POINT_ALREADY_EXISTS 1300
PRL_ERR_ISCSI_STORAGE_MOUNT_POINT_ALREADY_EXISTS macro 1300
PRL_ERR_ISCSI_STORAGE_MOUNTED 1300
PRL_ERR_ISCSI_STORAGE_MOUNTED macro 1300
PRL_ERR_ISCSI_STORAGE_NOT_FOUND 1300
PRL_ERR_ISCSI_STORAGE_NOT_FOUND macro 1300
PRL_ERR_ISCSI_STORAGE_NOT_MOUNTED 1301
PRL_ERR_ISCSI_STORAGE_NOT_MOUNTED macro 1301
PRL_ERR_ISCSI_STORAGE_NOT_SUPPORTED 1301
PRL_ERR_ISCSI_STORAGE_NOT_SUPPORTED macro 1301
PRL_ERR_ISCSI_STORAGE_REMOVE 1301
PRL_ERR_ISCSI_STORAGE_REMOVE macro 1301
PRL_ERR_ISCSI_STORAGE_START 1301
PRL_ERR_ISCSI_STORAGE_START macro 1301
PRL_ERR_ISCSI_STORAGE_UMOUNT 1301
PRL_ERR_ISCSI_STORAGE_UMOUNT macro 1301
PRL_ERR_IT_CANT_COMPACT 1302
PRL_ERR_IT_CANT_COMPACT macro 1302
PRL_ERR_IT_CANT_COMPACT_LDM_DISK 1302
PRL_ERR_IT_CANT_COMPACT_LDM_DISK macro 1302
PRL_ERR_IT_CANT_COMPACT_PLAIN_DISK 1302
PRL_ERR_IT_CANT_COMPACT_PLAIN_DISK macro 1302
PRL_ERR_IT_CANT_GET_BITMAP 1302
PRL_ERR_IT_CANT_GET_BITMAP macro 1302
PRL_ERR_IT_CANT_GET_PARAMS_FROM_OLD_IMAGE 1303
PRL_ERR_IT_CANT_GET_PARAMS_FROM_OLD_IMAGE macro 1303
PRL_ERR_IT_CANT_OPEN_IMAGE 1303
PRL_ERR_IT_CANT_OPEN_IMAGE macro 1303
PRL_ERR_IT_CANT_PREPARE_RECONFIG_DATA 1303
PRL_ERR_IT_CANT_PREPARE_RECONFIG_DATA macro 1303
PRL_ERR_IT_CANT_RECONFIG 1303
PRL_ERR_IT_CANT_RECONFIG macro 1303
PRL_ERR_IT_CANT_RECONFIG_BOOTCAMP 1303
PRL_ERR_IT_CANT_RECONFIG_BOOTCAMP macro 1303
PRL_ERR_IT_CANT_RESIZE_LDM_DISK 1304
PRL_ERR_IT_CANT_RESIZE_LDM_DISK macro 1304
PRL_ERR_IT_CANT_RESIZE_VOLUME 1304
PRL_ERR_IT_CANT_RESIZE_VOLUME macro 1304

PRL_ERR_IT_CANT_VALIDATE_BOOTCAMP 1304
PRL_ERR_IT_CANT_VALIDATE_BOOTCAMP macro 1304
PRL_ERR_IT_CONVERT_TO_CURRENT_ERROR 1304
PRL_ERR_IT_CONVERT_TO_CURRENT_ERROR macro 1304
PRL_ERR_IT_DISK_SMALL_FOR_SPLIT 1305
PRL_ERR_IT_DISK_SMALL_FOR_SPLIT macro 1305
PRL_ERR_IT_FAST_RESIZE_FAILURE 1305
PRL_ERR_IT_FAST_RESIZE_FAILURE macro 1305
PRL_ERR_IT_FIRST_ERROR 1305
PRL_ERR_IT_FIRST_ERROR macro 1305
PRL_ERR_IT_FIRST_FAILURE 1305
PRL_ERR_IT_FIRST_FAILURE macro 1305
PRL_ERR_IT_FS_NOT_SUPPORTED_FOR_RESIZE 1305
PRL_ERR_IT_FS_NOT_SUPPORTED_FOR_RESIZE macro 1305
PRL_ERR_IT_GUEST_TOOLS_UNKNOWN_STATE 1306
PRL_ERR_IT_GUEST_TOOLS_UNKNOWN_STATE macro 1306
PRL_ERR_IT_HAL_NOT_FOUND 1306
PRL_ERR_IT_HAL_NOT_FOUND macro 1306
PRL_ERR_IT_RECONFIG_PATH_NEEDED 1306
PRL_ERR_IT_RECONFIG_PATH_NEEDED macro 1306
PRL_ERR_IT_ROLLBACK_FAILURE 1306
PRL_ERR_IT_ROLLBACK_FAILURE macro 1306
PRL_ERR_IT_SIZE_CHANGED 1307
PRL_ERR_IT_SIZE_CHANGED macro 1307
PRL_ERR_IT_SNAPSHOTS_MERGE_IS_NEEDED 1307
PRL_ERR_IT_SNAPSHOTS_MERGE_IS_NEEDED macro 1307
PRL_ERR_IT_UNKNOWN_OS 1307
PRL_ERR_IT_UNKNOWN_OS macro 1307
PRL_ERR_IT_USER_INTERRUPTED 1307
PRL_ERR_IT_USER_INTERRUPTED macro 1307
PRL_ERR_LIC_REGISTRATION_COMMON_ERROR 1307
PRL_ERR_LIC_REGISTRATION_COMMON_ERROR macro 1307
PRL_ERR_LICENSE_AUTH_FAILED 1308
PRL_ERR_LICENSE_AUTH_FAILED macro 1308
PRL_ERR_LICENSE_BETA_KEY_RELEASE_PRODUCT 1308
PRL_ERR_LICENSE_BETA_KEY_RELEASE_PRODUCT macro 1308
PRL_ERR_LICENSE_BLACKLISTED 1308
PRL_ERR_LICENSE_BLACKLISTED macro 1308
PRL_ERR_LICENSE_BLACKLISTED_TO_VM_OPERATION 1308
PRL_ERR_LICENSE_BLACKLISTED_TO_VM_OPERATION macro 1308
PRL_ERR_LICENSE_DEFERRED_LICENSE_NOT_FOUND 1309
PRL_ERR_LICENSE_DEFERRED_LICENSE_NOT_FOUND macro 1309
PRL_ERR_LICENSE_EXPIRED 1309
PRL_ERR_LICENSE_EXPIRED macro 1309
PRL_ERR_LICENSE_FILE_WRITE_FAILED 1309
PRL_ERR_LICENSE_FILE_WRITE_FAILED macro 1309
PRL_ERR_LICENSE_GRACED 1309
PRL_ERR_LICENSE_GRACED macro 1309
PRL_ERR_LICENSE_IS_NOT_VOLUME 1309
PRL_ERR_LICENSE_IS_NOT_VOLUME macro 1309
PRL_ERR_LICENSE_NOT_STARTED 1310
PRL_ERR_LICENSE_NOT_STARTED macro 1310
PRL_ERR_LICENSE_NOT_VALID 1310
PRL_ERR_LICENSE_NOT_VALID macro 1310
PRL_ERR_LICENSE_RELEASE_KEY_BETA_PRODUCT 1310
PRL_ERR_LICENSE_RELEASE_KEY_BETA_PRODUCT macro 1310
PRL_ERR_LICENSE_RESTRICTED_CPU_COUNT 1310
PRL_ERR_LICENSE_RESTRICTED_CPU_COUNT macro 1310
PRL_ERR_LICENSE_RESTRICTED_GUEST_OS 1311
PRL_ERR_LICENSE_RESTRICTED_GUEST_OS macro 1311
PRL_ERR_LICENSE_RESTRICTED_TO_CLONE_VM 1311
PRL_ERR_LICENSE_RESTRICTED_TO_CLONE_VM macro 1311
PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_FROM_TEMPLATE 1311
PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_FROM_TEMPLATE macro 1311
PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_TO_TEMPLATE 1311
PRL_ERR_LICENSE_RESTRICTED_TO_CONVERT_TO_TEMPLATE macro 1311

PRL_ERR_LICENSE_RESTRICTED_TO_CREATE_VM 1312

PRL_ERR_LICENSE_RESTRICTED_TO_CREATE_VM macro 1312

PRL_ERR_LICENSE_RESTRICTED_TO_PAUSE_VM 1312

PRL_ERR_LICENSE_RESTRICTED_TO_PAUSE_VM macro 1312

PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_3RD_PARTY_VM 1312

PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_3RD_PARTY_VM macro 1312

PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_VM 1312

PRL_ERR_LICENSE_RESTRICTED_TO_REGISTER_VM macro 1312

PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT 1313

PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT macro 1313

PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT_PER_USER 1313

PRL_ERR_LICENSE_RESTRICTED_TO_RUNNING_VMS_LIMIT_PER_USER macro 1313

PRL_ERR_LICENSE_RESTRICTED_TO_SAFEMODE_FEATURE 1313

PRL_ERR_LICENSE_RESTRICTED_TO_SAFEMODE_FEATURE macro 1313

PRL_ERR_LICENSE_RESTRICTED_TO_SMARTGUARD_FEATURE 1314

PRL_ERR_LICENSE_RESTRICTED_TO_SMARTGUARD_FEATURE macro 1314

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_CREATE 1314

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_CREATE macro 1314

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_DELETE 1314

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_DELETE macro 1314

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SHOW_TREE 1315

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SHOW_TREE macro 1315

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SWITCH 1315

PRL_ERR_LICENSE_RESTRICTED_TO_SNAPSHOT_SWITCH macro 1315

PRL_ERR_LICENSE_RESTRICTED_TO_SUSPEND_VM 1315

PRL_ERR_LICENSE_RESTRICTED_TO_SUSPEND_VM macro 1315

PRL_ERR_LICENSE_RESTRICTED_TO_UNDODISK_FEATURE 1315

PRL_ERR_LICENSE_RESTRICTED_TO_UNDODISK_FEATURE macro 1315

PRL_ERR_LICENSE_TOO_MANY_MEMORY 1316

PRL_ERR_LICENSE_TOO_MANY_MEMORY macro 1316

PRL_ERR_LICENSE_TOO_MANY_VCPUS 1316

PRL_ERR_LICENSE_TOO_MANY_VCPUS macro 1316

PRL_ERR_LICENSE_UNSUPPORTED_LICENSE_TYPE_TO_DEACTIVATION 1316

PRL_ERR_LICENSE_UNSUPPORTED_LICENSE_TYPE_TO_DEACTIVATION macro 1316

PRL_ERR_LICENSE_UPGRADE_NO_ACCEPTABLE_LICENSE 1316

PRL_ERR_LICENSE_UPGRADE_NO_ACCEPTABLE_LICENSE macro 1316

PRL_ERR_LICENSE_UPGRADE_NO_PERMANENT_LICENSE 1317

PRL_ERR_LICENSE_UPGRADE_NO_PERMANENT_LICENSE macro 1317

PRL_ERR_LICENSE_VALID 1317

PRL_ERR_LICENSE_VALID macro 1317

PRL_ERR_LICENSE_VM_HAS_VTD_DEVICES 1317

PRL_ERR_LICENSE_VM_HAS_VTD_DEVICES macro 1317

PRL_ERR_LICENSE_WRONG_ADVANCED_FIELD 1317
PRL_ERR_LICENSE_WRONG_ADVANCED_FIELD macro 1317
PRL_ERR_LICENSE_WRONG_DISTRIBUTOR 1318
PRL_ERR_LICENSE_WRONG_DISTRIBUTOR macro 1318
PRL_ERR_LICENSE_WRONG_LANGUAGE 1318
PRL_ERR_LICENSE_WRONG_LANGUAGE macro 1318
PRL_ERR_LICENSE_WRONG_PLATFORM 1318
PRL_ERR_LICENSE_WRONG_PLATFORM macro 1318
PRL_ERR_LICENSE_WRONG_PRODUCT 1318
PRL_ERR_LICENSE_WRONG_PRODUCT macro 1318
PRL_ERR_LICENSE_WRONG_VERSION 1318
PRL_ERR_LICENSE_WRONG_VERSION macro 1318
PRL_ERR_LOCAL_AUTHENTICATION_FAILED 1319
PRL_ERR_LOCAL_AUTHENTICATION_FAILED macro 1319
PRL_ERR_LOGIN_BY_MISMATCH_CLIENT_MODE 1319
PRL_ERR_LOGIN_BY_MISMATCH_CLIENT_MODE macro 1319
PRL_ERR_LOW_MEMORY_LIMIT 1319
PRL_ERR_LOW_MEMORY_LIMIT macro 1319
PRL_ERR_LPT_DEVICE_ALREADY_EXIST 1319
PRL_ERR_LPT_DEVICE_ALREADY_EXIST macro 1319
PRL_ERR_LPT_OUTPUT_FILE_IS_NOT_SPECIFIED 1319
PRL_ERR_LPT_OUTPUT_FILE_IS_NOT_SPECIFIED macro 1319
PRL_ERR_LPT_OUTPUT_FILE_NOT_EXIST 1320
PRL_ERR_LPT_OUTPUT_FILE_NOT_EXIST macro 1320
PRL_ERR_MAC_ADDRESS_IN_CORRECT_LENGTH 1320
PRL_ERR_MAC_ADDRESS_IN_CORRECT_LENGTH macro 1320
PRL_ERR_MAC_ADDRESS_INCORRECT 1320
PRL_ERR_MAC_ADDRESS_INCORRECT macro 1320
PRL_ERR_MAC_ADDRESS_IS_EMPTY 1320
PRL_ERR_MAC_ADDRESS_IS_EMPTY macro 1320
PRL_ERR_MAC_ADDRESS_WITH_2_ZERO_START 1320
PRL_ERR_MAC_ADDRESS_WITH_2_ZERO_START macro 1320
PRL_ERR_MAC_ADDRESS_WITH_ALL_ZEROS 1321
PRL_ERR_MAC_ADDRESS_WITH_ALL_ZEROS macro 1321
PRL_ERR_MAC_ADDRESS_WITHIN_CORRECT_SYMBOLS 1321
PRL_ERR_MAC_ADDRESS_WITHIN_CORRECT_SYMBOLS macro 1321
PRL_ERR_MAKE_DIRECTORY 1321
PRL_ERR_MAKE_DIRECTORY macro 1321
PRL_ERR_MEM_EXCEED_PHY_SPACE 1321
PRL_ERR_MEM_EXCEED_PHY_SPACE macro 1321
PRL_ERR_MEMORY_ALLOC_ERROR 1321
PRL_ERR_MEMORY_ALLOC_ERROR macro 1321
PRL_ERR_MERGE_XML_DOCUMENT_CONFLICT 1322
PRL_ERR_MERGE_XML_DOCUMENT_CONFLICT macro 1322
PRL_ERR_MOBILE_ADVANCED_AUTH_REQUIRED 1322
PRL_ERR_MOBILE_ADVANCED_AUTH_REQUIRED macro 1322
PRL_ERR_MOUNTER_LIST_NO_OBJECT 1322
PRL_ERR_MOUNTER_LIST_NO_OBJECT macro 1322
PRL_ERR_MSG_START_32BIT_VM_ON_64BIT_HOST 1322
PRL_ERR_MSG_START_32BIT_VM_ON_64BIT_HOST macro 1322
PRL_ERR_NEED_KILL_PREV_SETTING 1323
PRL_ERR_NEED_KILL_PREV_SETTING macro 1323
PRL_ERR_NETWORK_ADAPTER_NOT_FOUND 1323
PRL_ERR_NETWORK_ADAPTER_NOT_FOUND macro 1323
PRL_ERR_NETWORK_ROLLBACK_FAILED 1323
PRL_ERR_NETWORK_ROLLBACK_FAILED macro 1323
PRL_ERR_NO_BOOTING_DEVICE_SELECTED 1323
PRL_ERR_NO_BOOTING_DEVICE_SELECTED macro 1323
PRL_ERR_NO_CD_DRIVE_AVAILABLE 1323
PRL_ERR_NO_CD_DRIVE_AVAILABLE macro 1323
PRL_ERR_NO_DATA 1324
PRL_ERR_NO_DATA macro 1324
PRL_ERR_NO_DISP_CONFIG_FOUND 1324
PRL_ERR_NO_DISP_CONFIG_FOUND macro 1324
PRL_ERR_NO_GUEST_OS_FOUND 1324
PRL_ERR_NO_GUEST_OS_FOUND macro 1324
PRL_ERR_NO_MORE_FREE_INTERFACE_SLOTS 1324
PRL_ERR_NO_MORE_FREE_INTERFACE_SLOTS macro 1324
PRL_ERR_NO_ONE_HARD_DISK_TO_COMPRESS 1324
PRL_ERR_NO_ONE_HARD_DISK_TO_COMPRESS macro 1324

PRL_ERR_NO_PROBLEM_REPORT_FOUND 1325
PRL_ERR_NO_PROBLEM_REPORT_FOUND macro 1325
PRL_ERR_NO_TARGET_DIR_PATH_SPECIFIED 1325
PRL_ERR_NO_TARGET_DIR_PATH_SPECIFIED macro 1325
PRL_ERR_NO_TARGET_PATH_SPECIFIED 1325
PRL_ERR_NO_TARGET_PATH_SPECIFIED macro 1325
PRL_ERR_NO_VM_DIR_CONFIG_FOUND 1325
PRL_ERR_NO_VM_DIR_CONFIG_FOUND macro 1325
PRL_ERR_NOT_ALL_FILES_WAS_DELETED 1325
PRL_ERR_NOT_ALL_FILES_WAS_DELETED macro 1325
PRL_ERR_NOT_CONNECTED_TO_DISPATCHER 1326
PRL_ERR_NOT_CONNECTED_TO_DISPATCHER macro 1326
PRL_ERR_NOT_CONNECTED_TO_PROXY_MANAGER 1326
PRL_ERR_NOT_CONNECTED_TO_PROXY_MANAGER macro 1326
PRL_ERR_NOT_ENOUGH_DISK_FREE_SPACE 1326
PRL_ERR_NOT_ENOUGH_DISK_FREE_SPACE macro 1326
PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_START_VM 1326
PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_START_VM macro 1326
PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_XML_SAVE 1327
PRL_ERR_NOT_ENOUGH_DISK_SPACE_TO_XML_SAVE macro 1327
PRL_ERR_NOT_ENOUGH_PERMS_TO_OPEN_AUTHORIZATION_FILE 1327
PRL_ERR_NOT_ENOUGH_PERMS_TO_OPEN_AUTHORIZATION_FILE macro 1327
PRL_ERR_NOT_ENOUGH_RIGHTS_FOR_LINKED_CLONE 1327
PRL_ERR_NOT_ENOUGH_RIGHTS_FOR_LINKED_CLONE macro 1327
PRL_ERR_NOT_LOCK_OWNER_SESSION_TRIES_TO_UNLOCK 1327
PRL_ERR_NOT_LOCK_OWNER_SESSION_TRIES_TO_UNLOCK macro 1327
PRL_ERR_NOT_SENTILLION_CLIENT 1328
PRL_ERR_NOT_SENTILLION_CLIENT macro 1328
PRL_ERR_NVRAM_FILE_COPY 1328
PRL_ERR_NVRAM_FILE_COPY macro 1328
PRL_ERR_OBJECT_BAD_INTERFACE 1328
PRL_ERR_OBJECT_BAD_INTERFACE macro 1328
PRL_ERR_OBJECT_CLASS_NOT_FOUND 1328
PRL_ERR_OBJECT_CLASS_NOT_FOUND macro 1328
PRL_ERR_OBJECT_DUPLICATE_CLASS 1328
PRL_ERR_OBJECT_DUPLICATE_CLASS macro 1328
PRL_ERR_OBJECT_DUPLICATE_UID 1329
PRL_ERR_OBJECT_DUPLICATE_UID macro 1329
PRL_ERR_OBJECT_LIB_CANT_GET_PERMS 1329
PRL_ERR_OBJECT_LIB_CANT_GET_PERMS macro 1329
PRL_ERR_OBJECT_LIB_LOAD_ERROR 1329
PRL_ERR_OBJECT_LIB_LOAD_ERROR macro 1329
PRL_ERR_OBJECT_LIB_NO_FUNCTIONS 1329
PRL_ERR_OBJECT_LIB_NO_FUNCTIONS macro 1329
PRL_ERR_OBJECT_LIB_WRONG_PERMS 1330
PRL_ERR_OBJECT_LIB_WRONG_PERMS macro 1330
PRL_ERR_OBJECT_NOT_FOUND 1330
PRL_ERR_OBJECT_NOT_FOUND macro 1330
PRL_ERR_OBJECT_WAS_REMOVED 1330
PRL_ERR_OBJECT_WAS_REMOVED macro 1330
PRL_ERR_ONLY_ADMIN_CAN_SET_PARAMETER_STARTLOGINMODE_ROOT 1330
PRL_ERR_ONLY_ADMIN_CAN_SET_PARAMETER_STARTLOGINMODE_ROOT macro 1330
PRL_ERR_ONLY_ADMIN_CAN_SET_VERBOSE_LOGGING 1331
PRL_ERR_ONLY_ADMIN_CAN_SET_VERBOSE_LOGGING macro 1331
PRL_ERR_ONLY_ADMIN_OR_VM_OWNER_CAN_OPEN_THIS_SESSION 1331
PRL_ERR_ONLY_ADMIN_OR_VM_OWNER_CAN_OPEN_THIS_SESSION macro 1331
PRL_ERR_OPEN_DISP_CONFIG_READ 1331
PRL_ERR_OPEN_DISP_CONFIG_READ macro 1331
PRL_ERR_OPEN_DISP_CONFIG_WRITE 1331
PRL_ERR_OPEN_DISP_CONFIG_WRITE macro 1331
PRL_ERR_OPEN_FAILED 1332
PRL_ERR_OPEN_FAILED macro 1332

PRL_ERR_OPEN_PROBLEM_REPORT_READ 1332
PRL_ERR_OPEN_PROBLEM_REPORT_READ macro 1332
PRL_ERR_OPEN_VM_CONFIG_READ 1332
PRL_ERR_OPEN_VM_CONFIG_READ macro 1332
PRL_ERR_OPEN_VM_CONFIG_WRITE 1332
PRL_ERR_OPEN_VM_CONFIG_WRITE macro 1332
PRL_ERR_OPEN_VM_DIR_CONFIG_READ 1332
PRL_ERR_OPEN_VM_DIR_CONFIG_READ macro 1332
PRL_ERR_OPEN_VM_DIR_CONFIG_WRITE 1333
PRL_ERR_OPEN_VM_DIR_CONFIG_WRITE macro 1333
PRL_ERR_OPERATION_FAILED 1333
PRL_ERR_OPERATION_FAILED macro 1333
PRL_ERR_OPERATION_PENDING 1333
PRL_ERR_OPERATION_PENDING macro 1333
PRL_ERR_OPERATION_WAS_CANCELED 1333
PRL_ERR_OPERATION_WAS_CANCELED macro 1333
PRL_ERR_OS_RECONFIG_DATA_ABSENT 1333
PRL_ERR_OS_RECONFIG_DATA_ABSENT macro 1333
PRL_ERR_OUT_OF_DISK_SPACE 1334
PRL_ERR_OUT_OF_DISK_SPACE macro 1334
PRL_ERR_OUT_OF_MEMORY 1334
PRL_ERR_OUT_OF_MEMORY macro 1334
PRL_ERR_PARALLEL_PORT_IMAGE_NOT_EXIST 1334
PRL_ERR_PARALLEL_PORT_IMAGE_NOT_EXIST macro 1334
PRL_ERR_PARALLEL_PORT_IMG_COPY 1334
PRL_ERR_PARALLEL_PORT_IMG_COPY macro 1334
PRL_ERR_PARALLEL_PORT_NO_RASTERIZER 1334
PRL_ERR_PARALLEL_PORT_NO_RASTERIZER macro 1334
PRL_ERR_PARAM_NOT_FOUND 1335
PRL_ERR_PARAM_NOT_FOUND macro 1335
PRL_ERR_PARSE_CLIENT_PREFS 1335
PRL_ERR_PARSE_CLIENT_PREFS macro 1335
PRL_ERR_PARSE_COMMON_SERVER_PREFS 1335
PRL_ERR_PARSE_COMMON_SERVER_PREFS macro 1335
PRL_ERR_PARSE_DISP_CONFIG 1335
PRL_ERR_PARSE_DISP_CONFIG macro 1335
PRL_ERR_PARSE_FILESYSTEM_INFO 1335
PRL_ERR_PARSE_FILESYSTEM_INFO macro 1335
PRL_ERR_PARSE_HARD_DISK_HW_INFO 1336
PRL_ERR_PARSE_HARD_DISK_HW_INFO macro 1336
PRL_ERR_PARSE_HOST_HW_INFO 1336
PRL_ERR_PARSE_HOST_HW_INFO macro 1336
PRL_ERR_PARSE_PROBLEM_REPORT 1336
PRL_ERR_PARSE_PROBLEM_REPORT macro 1336
PRL_ERR_PARSE_STATISTICS 1336
PRL_ERR_PARSE_STATISTICS macro 1336
PRL_ERR_PARSE_USER_PROFILE 1336
PRL_ERR_PARSE_USER_PROFILE macro 1336
PRL_ERR_PARSE_VM_CONFIG 1337
PRL_ERR_PARSE_VM_CONFIG macro 1337
PRL_ERR_PARSE_VM_DIR_CONFIG 1337
PRL_ERR_PARSE_VM_DIR_CONFIG macro 1337
PRL_ERR_PARSING_EVENT 1337
PRL_ERR_PARSING_EVENT macro 1337
PRL_ERR_PATH_IS_NOT_DIRECTORY 1337
PRL_ERR_PATH_IS_NOT_DIRECTORY macro 1337
PRL_ERR_PAX_ACCESS_FORBIDDEN 1337
PRL_ERR_PAX_ACCESS_FORBIDDEN macro 1337
PRL_ERR_PAX_HOST_LIMIT_WAS_EXCEEDED 1338
PRL_ERR_PAX_HOST_LIMIT_WAS_EXCEEDED macro 1338
PRL_ERR_PCMOVER_EXEC_FAILED 1338
PRL_ERR_PCMOVER_EXEC_FAILED macro 1338
PRL_ERR_PCMOVER_MIGRATE_FAILED 1338
PRL_ERR_PCMOVER_MIGRATE_FAILED macro 1338
PRL_ERR_PCMOVER_NOT_INSTALLED 1338
PRL_ERR_PCMOVER_NOT_INSTALLED macro 1338
PRL_ERR_PCMOVER_POLICY_FILE_OPEN_ERROR 1338
PRL_ERR_PCMOVER_POLICY_FILE_OPEN_ERROR macro 1338
PRL_ERR_PCMOVER_POLICY_FILE_WRITE_ERROR 1339
PRL_ERR_PCMOVER_POLICY_FILE_WRITE_ERROR macro 1339
PRL_ERR_PCMOVER_VAN_FILE_CREATE_ERROR 1339
PRL_ERR_PCMOVER_VAN_FILE_CREATE_ERROR macro 1339
PRL_ERR_PCMOVER_VAN_FILE_OPEN_ERROR 1339
PRL_ERR_PCMOVER_VAN_FILE_OPEN_ERROR macro 1339
PRL_ERR_PCMOVER_VAN_FILE_PREPARE_FAILED 1339
PRL_ERR_PCMOVER_VAN_FILE_PREPARE_FAILED macro 1339

PRL_ERR_PCMOVER_WINDOWS_DIR_NOT_EXIST 1340
PRL_ERR_PCMOVER_WINDOWS_DIR_NOT_EXIST macro 1340
PRL_ERR_PERFORM_MOUNTER_COMMAND 1340
PRL_ERR_PERFORM_MOUNTER_COMMAND macro 1340
PRL_ERR_PLAYER_CANT_SUSPEND_IN_PAUSE 1340
PRL_ERR_PLAYER_CANT_SUSPEND_IN_PAUSE macro 1340
PRL_ERR_PPC_INVALID_CREDENTIALS 1340
PRL_ERR_PPC_INVALID_CREDENTIALS macro 1340
PRL_ERR_PPC_INVALID_CREDENTIALS_ON_RECONNECT 1341
PRL_ERR_PPC_INVALID_CREDENTIALS_ON_RECONNECT macro 1341
PRL_ERR_PPC_SERVER_BUSY 1341
PRL_ERR_PPC_SERVER_BUSY macro 1341
PRL_ERR_PPC_UNABLE_TO_CONNECT 1341
PRL_ERR_PPC_UNABLE_TO_CONNECT macro 1341
PRL_ERR_PREPARE_FOR_HIBERNATE_TASK_ALREADY_RUN 1341
PRL_ERR_PREPARE_FOR_HIBERNATE_TASK_ALREADY_RUN macro 1341
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_CANNOT_STAND_BY 1342
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_CANNOT_STAND_BY macro 1342
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WITHOUT_TOOLS 1342
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WITHOUT_TOOLS macro 1342
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WRONG_STATE 1342
PRL_ERR_PREPARE_FOR_HIBERNATE_VM_WRONG_STATE macro 1342
PRL_ERR_PREV_SESSION_IS_ACTIVE 1342
PRL_ERR_PREV_SESSION_IS_ACTIVE macro 1342
PRL_ERR_PROBLEM_REPORT_ALREADY_EXISTS 1343
PRL_ERR_PROBLEM_REPORT_ALREADY_EXISTS macro 1343
PRL_ERR_PROBLEM_REPORT_FILE_NOTSET 1343
PRL_ERR_PROBLEM_REPORT_FILE_NOTSET macro 1343
PRL_ERR_PROBLEM_REPORT_WRITE 1343
PRL_ERR_PROBLEM_REPORT_WRITE macro 1343
PRL_ERR_PROXY_HANDSHAKE_FAILED 1343
PRL_ERR_PROXY_HANDSHAKE_FAILED macro 1343
PRL_ERR_PROXY_PEER_NOT_FOUND 1343
PRL_ERR_PROXY_PEER_NOT_FOUND macro 1343
PRL_ERR_PROXY_WRONG_PORT_NUMBER 1344
PRL_ERR_PROXY_WRONG_PORT_NUMBER macro 1344
PRL_ERR_PROXY_WRONG_PROTOCOL_VERSION 1344
PRL_ERR_PROXY_WRONG_PROTOCOL_VERSION macro 1344
PRL_ERR_READ_FAILED 1344
PRL_ERR_READ_FAILED macro 1344
PRL_ERR_READ_XML_CONTENT 1344
PRL_ERR_READ_XML_CONTENT macro 1344
PRL_ERR_READONLY_FILESYSTEM 1344
PRL_ERR_READONLY_FILESYSTEM macro 1344
PRL_ERR_REBOOT_HOST 1345
PRL_ERR_REBOOT_HOST macro 1345
PRL_ERR_REG_PSTORAGE_REVOKE_IS_NEEDS 1345
PRL_ERR_REG_PSTORAGE_REVOKE_IS_NEEDS macro 1345
PRL_ERR_REMOTE_DEVICE_EXIST 1345
PRL_ERR_REMOTE_DEVICE_EXIST macro 1345
PRL_ERR_REMOTE_DEVICE_NOT_EXIST 1345
PRL_ERR_REMOTE_DEVICE_NOT_EXIST macro 1345
PRL_ERR_REMOTE_DEVICE_NOT_REGISTERED 1345
PRL_ERR_REMOTE_DEVICE_NOT_REGISTERED macro 1345
PRL_ERR_REMOTE_DISPLAY_EMPTY_PASSWORD 1346
PRL_ERR_REMOTE_DISPLAY_EMPTY_PASSWORD macro 1346
PRL_ERR_REMOTE_DISPLAY_HOST_NOT_SPECIFIED 1346
PRL_ERR_REMOTE_DISPLAY_HOST_NOT_SPECIFIED macro 1346
PRL_ERR_REMOTE_DISPLAY_WRONG_PORT_NUMBER 1346
PRL_ERR_REMOTE_DISPLAY_WRONG_PORT_NUMBER macro 1346
PRL_ERR_RESUME_BOOTCAMP_CHANGED_DISK_CONTE

NTS
1346
PRL_ERR_RESUME_BOOTCAMP_CHANGED_DISK_CONTENTS
macro 1346
PRL_ERR_RESUME_BOOTCAMP_CORRUPT_DISK_STATE_PARAM
1347
PRL_ERR_RESUME_BOOTCAMP_CORRUPT_DISK_STATE_PARAM
macro 1347
PRL_ERR_RETRIEVE_VM_CONFIG 1347
PRL_ERR_RETRIEVE_VM_CONFIG macro 1347
PRL_ERR_RETURN_CODE_RANG_EEND 1347
PRL_ERR_RETURN_CODE_RANG_EEND macro 1347
PRL_ERR_REVERT_IMPERSONATE_FAILED 1347
PRL_ERR_REVERT_IMPERSONATE_FAILED macro 1347
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT
1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_BY_GUEST_SLEEP_TIMEOUT
macro 1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_PAUSED 1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_PAUSED macro 1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL
1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL
macro 1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL
1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL
macro 1348
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS
1349
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_CAPS
macro 1349
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL
1349
PRL_ERR_REVERT_SNAPSHOT_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL
macro 1349
PRL_ERR_RUN_VM_ACTION_SCRIPT 1349
PRL_ERR_RUN_VM_ACTION_SCRIPT macro 1349
PRL_ERR_RUNNING_VM_OR_CT 1349
PRL_ERR_RUNNING_VM_OR_CT macro 1349
PRL_ERR_SAFE_MODE_START_DURING_SUSPENDING_SYNC
1350
PRL_ERR_SAFE_MODE_START_DURING_SUSPENDING_SYNC
macro 1350
PRL_ERR_SAMPLE_CONFIG_NOT_FOUND 1350
PRL_ERR_SAMPLE_CONFIG_NOT_FOUND macro 1350
PRL_ERR_SAVE_VM_CATALOG 1350
PRL_ERR_SAVE_VM_CATALOG macro 1350
PRL_ERR_SAVE_VM_CONFIG 1350
PRL_ERR_SAVE_VM_CONFIG macro 1350
PRL_ERR_SDK_TRY_AGAIN 1351
PRL_ERR_SDK_TRY_AGAIN macro 1351
PRL_ERR_SEARCH_CONFIG_OPERATION_CANCELED
1351
PRL_ERR_SEARCH_CONFIG_OPERATION_CANCELED
macro 1351
PRL_ERR_SEND_COMMAND_TOWS_FAILED 1351
PRL_ERR_SEND_COMMAND_TOWS_FAILED macro 1351
PRL_ERR_SERIAL_IMG_COPY 1351
PRL_ERR_SERIAL_IMG_COPY macro 1351
PRL_ERR_SERIAL_IMG_NOT_FOUND 1352
PRL_ERR_SERIAL_IMG_NOT_FOUND macro 1352
PRL_ERR_SERIAL_PORT_IMAGE_NOT_EXIST 1352
PRL_ERR_SERIAL_PORT_IMAGE_NOT_EXIST macro 1352
PRL_ERR_SERVER_GOES_DOWN 1352
PRL_ERR_SERVER_GOES_DOWN macro 1352
PRL_ERR_SERVER_PREFS_EDIT_COLLISION 1352
PRL_ERR_SERVER_PREFS_EDIT_COLLISION macro 1352
PRL_ERR_SERVICE_BUSY 1352
PRL_ERR_SERVICE_BUSY macro 1352
PRL_ERR_SET_CPULIMIT 1353
PRL_ERR_SET_CPULIMIT macro 1353
PRL_ERR_SET_CPUMASK 1353
PRL_ERR_SET_CPUMASK macro 1353

PRL_ERR_SET_CPUUNITS 1353

PRL_ERR_SET_CPUUNITS macro 1353

PRL_ERR_SET_IOLIMIT 1353

PRL_ERR_SET_IOLIMIT macro 1353

PRL_ERR_SET_IOPRIO 1353

PRL_ERR_SET_IOPRIO macro 1353

PRL_ERR_SET_LICENSE_INVALID_ARG 1354

PRL_ERR_SET_LICENSE_INVALID_ARG macro 1354

PRL_ERR_SET_LICENSE_INVALID_KEY 1354

PRL_ERR_SET_LICENSE_INVALID_KEY macro 1354

PRL_ERR_SET_NETWORK_SETTINGS_FAILED 1354

PRL_ERR_SET_NETWORK_SETTINGS_FAILED macro 1354

PRL_ERR_SHUT_DOWN_NOT_IFICATION_SERVICE 1354

PRL_ERR_SHUT_DOWN_NOT_IFICATION_SERVICE macro 1354

PRL_ERR_SMC_ERROR 1355

PRL_ERR_SMC_ERROR macro 1355

PRL_ERR_SMP_NOT_SUPPORTED_HVT_DISABLED 1355

PRL_ERR_SMP_NOT_SUPPORTED_HVT_DISABLED macro 1355

PRL_ERR_SMP_NOT_SUPPORTED_HVT_NOT_PRESENT 1355

PRL_ERR_SMP_NOT_SUPPORTED_HVT_NOT_PRESENT macro 1355

PRL_ERR_SNAPSHOTS_COPY 1355

PRL_ERR_SNAPSHOTS_COPY macro 1355

PRL_ERR_SOME_TASKS_PRESENT 1355

PRL_ERR_SOME_TASKS_PRESENT macro 1355

PRL_ERR_SOME_VMS_RUNNING 1356

PRL_ERR_SOME_VMS_RUNNING macro 1356

PRL_ERR_SOUND_BAD_EMULATION_TYPE 1356

PRL_ERR_SOUND_BAD_EMULATION_TYPE macro 1356

PRL_ERR_SOUND_DEVICE_WRITE_FAILED 1356

PRL_ERR_SOUND_DEVICE_WRITE_FAILED macro 1356

PRL_ERR_SOUND_IN_DEVICE_OPEN_FAILED 1356

PRL_ERR_SOUND_IN_DEVICE_OPEN_FAILED macro 1356

PRL_ERR_SOUND_OUT_DEVICE_OPEN_FAILED 1357

PRL_ERR_SOUND_OUT_DEVICE_OPEN_FAILED macro 1357

PRL_ERR_SOURCE_ISNT_BOOTCAMP_DISK 1357

PRL_ERR_SOURCE_ISNT_BOOTCAMP_DISK macro 1357

PRL_ERR_SSL_HANDSHAKE_FAILED 1357

PRL_ERR_SSL_HANDSHAKE_FAILED macro 1357

PRL_ERR_STAND_BY_VM_BY_GUEST_SLEEP_TIMEOUT 1357

PRL_ERR_STAND_BY_VM_BY_GUEST_SLEEP_TIMEOUT macro 1357

PRL_ERR_STAND_BY_VM_PAUSED 1357

PRL_ERR_STAND_BY_VM_PAUSED macro 1357

PRL_ERR_STAND_BY_VM_WITH_OUTDATED_SHUTDOWN_TOOL 1358

PRL_ERR_STAND_BY_VM_WITH_OUTDATED_SHUTDOWN_TOOL macro 1358

PRL_ERR_STAND_BY_VM_WITH_UNLOADED_SHUTDOWN_TOOL 1358

PRL_ERR_STAND_BY_VM_WITH_UNLOADED_SHUTDOWN_TOOL macro 1358

PRL_ERR_STAND_BY_VM_WITH_UNSUPPORTED_SHUTDOWN_TOOL 1358

PRL_ERR_STAND_BY_VM_WITH_UNSUPPORTED_SHUTDOWN_TOOL macro 1358

PRL_ERR_START_SAFE_MODE_UNSUPPORTED 1359

PRL_ERR_START_SAFE_MODE_UNSUPPORTED macro 1359

PRL_ERR_START_VM_BY_NOT_AUTH_USER 1359

PRL_ERR_START_VM_BY_NOT_AUTH_USER macro 1359

PRL_ERR_START_VM_BY_NOT_DEFINED_USER 1359

PRL_ERR_START_VM_BY_NOT_DEFINED_USER macro 1359

PRL_ERR_STATE_ALREADY_EXISTS 1359

PRL_ERR_STATE_ALREADY_EXISTS macro 1359

PRL_ERR_STATE_CANT_LOAD_SPECIFIED_CFG 1359

PRL_ERR_STATE_CANT_LOAD_SPECIFIED_CFG macro 1359

PRL_ERR_STATE_CANT_OPEN_FOR_WRITE 1360

PRL_ERR_STATE_CANT_OPEN_FOR_WRITE macro 1360

PRL_ERR_STATE_CANT_OPEN_IMAGE 1360

PRL_ERR_STATE_CANT_OPEN_IMAGE macro 1360

PRL_ERR_STATE_CANT_OPEN_LOCKED 1360

PRL_ERR_STATE_CANT_OPEN_LOCKED macro 1360

PRL_ERR_STATE_CANT_SAVE_LOCKED 1360
PRL_ERR_STATE_CANT_SAVE_LOCKED macro 1360
PRL_ERR_STATE_DELETE_NONCLOSED 1360
PRL_ERR_STATE_DELETE_NONCLOSED macro 1360
PRL_ERR_STATE_ERROR_CREATE_IMAGE 1361
PRL_ERR_STATE_ERROR_CREATE_IMAGE macro 1361
PRL_ERR_STATE_ERROR_RENAMING_IMAGE 1361
PRL_ERR_STATE_ERROR_RENAMING_IMAGE macro 1361
PRL_ERR_STATE_FULL_DELETE_FAILED 1361
PRL_ERR_STATE_FULL_DELETE_FAILED macro 1361
PRL_ERR_STATE_GETFREESPACE_FAILED 1361
PRL_ERR_STATE_GETFREESPACE_FAILED macro 1361
PRL_ERR_STATE_INT_CORRUPTED 1361
PRL_ERR_STATE_INT_CORRUPTED macro 1361
PRL_ERR_STATE_INVALID_IMAGE_TYPE 1362
PRL_ERR_STATE_INVALID_IMAGE_TYPE macro 1362
PRL_ERR_STATE_INVALID_PARAMETERS 1362
PRL_ERR_STATE_INVALID_PARAMETERS macro 1362
PRL_ERR_STATE_MEMORY_ERROR 1362
PRL_ERR_STATE_MEMORY_ERROR macro 1362
PRL_ERR_STATE_MERGE_FAILED 1362
PRL_ERR_STATE_MERGE_FAILED macro 1362
PRL_ERR_STATE_MERGE_NO_SPACE 1362
PRL_ERR_STATE_MERGE_NO_SPACE macro 1362
PRL_ERR_STATE_NO_DISKS 1363
PRL_ERR_STATE_NO_DISKS macro 1363
PRL_ERR_STATE_NO_STATE 1363
PRL_ERR_STATE_NO_STATE macro 1363
PRL_ERR_STATE_NOT_OPENED 1363
PRL_ERR_STATE_NOT_OPENED macro 1363
PRL_ERR_STATE_NOT_PERMITTED 1363
PRL_ERR_STATE_NOT_PERMITTED macro 1363
PRL_ERR_STATE_NOT_RELEASED 1363
PRL_ERR_STATE_NOT_RELEASED macro 1363
PRL_ERR_STATE_PROCESS_RUNNING 1364
PRL_ERR_STATE_PROCESS_RUNNING macro 1364
PRL_ERR_STATE_ROLLBACK_ERROR 1364
PRL_ERR_STATE_ROLLBACK_ERROR macro 1364
PRL_ERR_STATE_ROLLBACK_IN_PROGRESS 1364
PRL_ERR_STATE_ROLLBACK_IN_PROGRESS macro 1364
PRL_ERR_STATE_STATFS_FAILED 1364
PRL_ERR_STATE_STATFS_FAILED macro 1364
PRL_ERR_STATE_STOPPING_STATE 1364
PRL_ERR_STATE_STOPPING_STATE macro 1364
PRL_ERR_STATE_UNEXPECTED_ERROR 1365
PRL_ERR_STATE_UNEXPECTED_ERROR macro 1365
PRL_ERR_SUCCESS 1365
PRL_ERR_SUCCESS macro 1365
PRL_ERR_SUSPEND_BOOTCAMP_NOT_NTFS_ONLY_DISK 1365
PRL_ERR_SUSPEND_BOOTCAMP_NOT_NTFS_ONLY_DISK macro 1365
PRL_ERR_SUSPEND_BOOTCAMP_NTFS_RW_MOUNTERS_DETECTED 1365
PRL_ERR_SUSPEND_BOOTCAMP_NTFS_RW_MOUNTERS_DETECTED macro 1365
PRL_ERR_SUSPEND_REJECTED_BY_GUEST 1366
PRL_ERR_SUSPEND_REJECTED_BY_GUEST macro 1366
PRL_ERR_SUSPEND_VM_VTD_BY_GUEST_SLEEP_TIMEOUT 1366
PRL_ERR_SUSPEND_VM_VTD_BY_GUEST_SLEEP_TIMEOUT macro 1366
PRL_ERR_SUSPEND_VM_VTD_PAUSED 1366
PRL_ERR_SUSPEND_VM_VTD_PAUSED macro 1366
PRL_ERR_SUSPEND_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL 1366
PRL_ERR_SUSPEND_VM_VTD_WITH_OUTDATED_SHUTDOWN_TOOL macro 1366
PRL_ERR_SUSPEND_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL 1367
PRL_ERR_SUSPEND_VM_VTD_WITH_UNLOADED_SHUTDOWN_TOOL macro 1367
PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_CAPS 1367
PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_CAPS macro 1367

PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL 1367
PRL_ERR_SUSPEND_VM_VTD_WITH_UNSUPPORTED_SHUTDOWN_TOOL macro 1367
PRL_ERR_SUSPEND_WITH_USB_BOOTDISK 1367
PRL_ERR_SUSPEND_WITH_USB_BOOTDISK macro 1367
PRL_ERR_SYMBOL_NOT_FOUND 1368
PRL_ERR_SYMBOL_NOT_FOUND macro 1368
PRL_ERR_TARGET_NAME_ALREADY_OCCUPIED 1368
PRL_ERR_TARGET_NAME_ALREADY_OCCUPIED macro 1368
PRL_ERR_TARGET_PATH_IS_NOT_DIRECTORY 1368
PRL_ERR_TARGET_PATH_IS_NOT_DIRECTORY macro 1368
PRL_ERR_TASK_NOT_FOUND 1368
PRL_ERR_TASK_NOT_FOUND macro 1368
PRL_ERR_TEMPLATE_HAS_APPS 1368
PRL_ERR_TEMPLATE_HAS_APPS macro 1368
PRL_ERR_TEMPLATE_NOT_FOUND 1369
PRL_ERR_TEMPLATE_NOT_FOUND macro 1369
PRL_ERR_TEST_TEXT_MESSAGES 1369
PRL_ERR_TEST_TEXT_MESSAGES macro 1369
PRL_ERR_TEST_TEXT_MESSAGES_PS 1369
PRL_ERR_TEST_TEXT_MESSAGES_PS macro 1369
PRL_ERR_TIME_MACHINE_EXCLUDED_LIST_OP 1369
PRL_ERR_TIME_MACHINE_EXCLUDED_LIST_OP macro 1369
PRL_ERR_TIMEOUT 1370
PRL_ERR_TIMEOUT macro 1370
PRL_ERR_TIS_INVALID_UID 1370
PRL_ERR_TIS_INVALID_UID macro 1370
PRL_ERR_TOO_LOW_HDD_SIZE 1370
PRL_ERR_TOO_LOW_HDD_SIZE macro 1370
PRL_ERR_TOOLS_UNSUPPORTED_GUEST 1370
PRL_ERR_TOOLS_UNSUPPORTED_GUEST macro 1370
PRL_ERR_TRY_AGAIN 1370
PRL_ERR_TRY_AGAIN macro 1370
PRL_ERR_UNABLE_APPLY_MEMORY_GUARANTEE 1371
PRL_ERR_UNABLE_APPLY_MEMORY_GUARANTEE macro 1371
PRL_ERR_UNABLE_APPLY_TOTAL_LIMIT 1371
PRL_ERR_UNABLE_APPLY_TOTAL_LIMIT macro 1371
PRL_ERR_UNABLE_DROP_SUSPENDED_STATE 1371
PRL_ERR_UNABLE_DROP_SUSPENDED_STATE macro 1371
PRL_ERR_UNABLE_SEND_REQUEST 1371
PRL_ERR_UNABLE_SEND_REQUEST macro 1371
PRL_ERR_UNABLE_TO_CONTINUE_VM_LIFETIME_IS_EXPIRED 1372
PRL_ERR_UNABLE_TO_CONTINUE_VM_LIFETIME_IS_EXPIRED macro 1372
PRL_ERR_UNABLE_TO_SEND_REQUEST 1372
PRL_ERR_UNABLE_TO_SEND_REQUEST macro 1372
PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE 1372
PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE macro 1372
PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE_BY_NON_PRIVILEGED_USER 1372
PRL_ERR_UNABLE_TO_SETUP_HEADLESS_MODE_BY_NON_PRIVILEGED_USER macro 1372
PRL_ERR_UNATTENDED_UNSUPPORTED_GUEST 1373
PRL_ERR_UNATTENDED_UNSUPPORTED_GUEST macro 1373
PRL_ERR_UNDER_OLD_HYPERVISOR 1373
PRL_ERR_UNDER_OLD_HYPERVISOR macro 1373
PRL_ERR_UNEXPECTED 1373
PRL_ERR_UNEXPECTED macro 1373
PRL_ERR_UNIMPLEMENTED 1373
PRL_ERR_UNIMPLEMENTED macro 1373
PRL_ERR_UNINITIALIZED 1373
PRL_ERR_UNINITIALIZED macro 1373
PRL_ERR_UNNAMED_CT_MOVE 1374
PRL_ERR_UNNAMED_CT_MOVE macro 1374
PRL_ERR_UNRECOGNIZED_REQUEST 1374
PRL_ERR_UNRECOGNIZED_REQUEST macro 1374
PRL_ERR_UNSUPPORTED_DEVICE_TYPE 1374
PRL_ERR_UNSUPPORTED_DEVICE_TYPE macro 1374
PRL_ERR_UNSUPPORTED_FILE_SYSTEM 1374
PRL_ERR_UNSUPPORTED_FILE_SYSTEM macro 1374
PRL_ERR_UNSUPPORTED_LAYOUTS_STRUCTURE 1374

PRL_ERR_UNSUPPORTED_LAYOUTS_STRUCTURE macro 1374
PRL_ERR_UNSUPPORTED_NETWORK_FILE_SYSTEM 1375
PRL_ERR_UNSUPPORTED_NETWORK_FILE_SYSTEM macro 1375
PRL_ERR_UNSUPPORTED_VIRTUAL_SOURCE 1375
PRL_ERR_UNSUPPORTED_VIRTUAL_SOURCE macro 1375
PRL_ERR_UPD_TOOLS_VER_VM_NOT_CONFIGURED 1375
PRL_ERR_UPD_TOOLS_VER_VM_NOT_CONFIGURED macro 1375
PRL_ERR_UPD_TOOLS_VER_VM_NOT_CREATED 1375
PRL_ERR_UPD_TOOLS_VER_VM_NOT_CREATED macro 1375
PRL_ERR_UPD_UPDATER_CONFIG 1375
PRL_ERR_UPD_UPDATER_CONFIG macro 1375
PRL_ERR_UPD_UPDATES 1376
PRL_ERR_UPD_UPDATES macro 1376
PRL_ERR_UPDATE_MEM_VM_NOT_CREATED 1376
PRL_ERR_UPDATE_MEM_VM_NOT_CREATED macro 1376
PRL_ERR_USER_CANT_CHANGE_ACCESS_PROFILE 1376
PRL_ERR_USER_CANT_CHANGE_ACCESS_PROFILE macro 1376
PRL_ERR_USER_CANT_CHANGE_PROFILE_ACCESS_PART 1376
PRL_ERR_USER_CANT_CHANGE_PROFILE_ACCESS_PART macro 1376
PRL_ERR_USER_CANT_CHANGE_READ_ONLY_VALUE 1377
PRL_ERR_USER_CANT_CHANGE_READ_ONLY_VALUE macro 1377
PRL_ERR_USER_DIRECTORY_NOT_SET 1377
PRL_ERR_USER_DIRECTORY_NOT_SET macro 1377
PRL_ERR_USER_IS_ALREADY_LOGGED 1377
PRL_ERR_USER_IS_ALREADY_LOGGED macro 1377
PRL_ERR_USER_NO_AUTH_TO_CREATE_ROOT_VM_DIR 1377
PRL_ERR_USER_NO_AUTH_TO_CREATE_ROOT_VM_DIR macro 1377
PRL_ERR_USER_NO_AUTH_TO_CREATE_VM_IN_DIR 1378
PRL_ERR_USER_NO_AUTH_TO_CREATE_VM_IN_DIR macro 1378
PRL_ERR_USER_NO_AUTH_TO_EDIT_SERVER_SETTINGS 1378
PRL_ERR_USER_NO_AUTH_TO_EDIT_SERVER_SETTINGS macro 1378
PRL_ERR_USER_NO_AUTH_TO_EDIT_VM 1378
PRL_ERR_USER_NO_AUTH_TO_EDIT_VM macro 1378
PRL_ERR_USER_NO_AUTH_TO_SAVE_BACKUP_FILES 1378
PRL_ERR_USER_NO_AUTH_TO_SAVE_BACKUP_FILES macro 1378
PRL_ERR_USER_NO_AUTH_TO_SAVE_FILES 1379
PRL_ERR_USER_NO_AUTH_TO_SAVE_FILES macro 1379
PRL_ERR_USER_NOT_FOUND 1379
PRL_ERR_USER_NOT_FOUND macro 1379
PRL_ERR_USER_OPERATION_NOT_AUTHORISED 1379
PRL_ERR_USER_OPERATION_NOT_AUTHORISED macro 1379
PRL_ERR_USER_PROFILE_WAS_CHANGED 1379
PRL_ERR_USER_PROFILE_WAS_CHANGED macro 1379
PRL_ERR_VA_CONFIG 1379
PRL_ERR_VA_CONFIG macro 1379
PRL_ERR_VCMMD_NO_CONNECTION 1380
PRL_ERR_VCMMD_NO_CONNECTION macro 1380
PRL_ERR_VM_ABORT 1380
PRL_ERR_VM_ABORT macro 1380
PRL_ERR_VM_ALLOC_MEM_DRV_NOT_STARTED 1380
PRL_ERR_VM_ALLOC_MEM_DRV_NOT_STARTED macro 1380
PRL_ERR_VM_ALLOC_VM_MEMORY 1380
PRL_ERR_VM_ALLOC_VM_MEMORY macro 1380
PRL_ERR_VM_ALREADY_CONNECTED 1381
PRL_ERR_VM_ALREADY_CONNECTED macro 1381
PRL_ERR_VM_ALREADY_CREATED 1381
PRL_ERR_VM_ALREADY_CREATED macro 1381
PRL_ERR_VM_ALREADY_REGISTERED 1381
PRL_ERR_VM_ALREADY_REGISTERED macro 1381
PRL_ERR_VM_ALREADY_REGISTERED_UNIQUE_PARAMS 1381
PRL_ERR_VM_ALREADY_REGISTERED_UNIQUE_PARAMS macro 1381
PRL_ERR_VM_ALREADY_REGISTERED_VM_NAME 1381
PRL_ERR_VM_ALREADY_REGISTERED_VM_NAME macro 1381
PRL_ERR_VM_ALREADY_REGISTERED_VM_PATH 1382

PRL_ERR_VM_ALREADY_REGISTERED_VM_PATH macro 1382

PRL_ERR_VM_ALREADY_REGISTERED_VM_UUID 1382

PRL_ERR_VM_ALREADY_REGISTERED_VM_UUID macro 1382

PRL_ERR_VM_ALREADY_RESUMED 1382

PRL_ERR_VM_ALREADY_RESUMED macro 1382

PRL_ERR_VM_ALREADY_RUNNING 1382

PRL_ERR_VM_ALREADY_RUNNING macro 1382

PRL_ERR_VM_ALREADY_SUSPENDED 1383

PRL_ERR_VM_ALREADY_SUSPENDED macro 1383

PRL_ERR_VM_ANOTHER_TOOLS_IN_USE 1383

PRL_ERR_VM_ANOTHER_TOOLS_IN_USE macro 1383

PRL_ERR_VM_APPLY_CHANGES_PENDING 1383

PRL_ERR_VM_APPLY_CHANGES_PENDING macro 1383

PRL_ERR_VM_APPLY_CONFIG_FAILED 1383

PRL_ERR_VM_APPLY_CONFIG_FAILED macro 1383

PRL_ERR_VM_APPLY_CONFIG_NEEDS_REBOOT 1383

PRL_ERR_VM_APPLY_CONFIG_NEEDS_REBOOT macro 1383

PRL_ERR_VM_ASYNC_CD_CONSTRUCTOR 1384

PRL_ERR_VM_ASYNC_CD_CONSTRUCTOR macro 1384

PRL_ERR_VM_AUTO_COMPRESS_TASK_ALREADY_RUN 1384

PRL_ERR_VM_AUTO_COMPRESS_TASK_ALREADY_RUN macro 1384

PRL_ERR_VM_BACKUP_HDD_IMAGE_OUT_OF_BUNDLE 1384

PRL_ERR_VM_BACKUP_HDD_IMAGE_OUT_OF_BUNDLE macro 1384

PRL_ERR_VM_BACKUP_INVALID_DISK_TYPE 1384

PRL_ERR_VM_BACKUP_INVALID_DISK_TYPE macro 1384

PRL_ERR_VM_BAD_OS_TYPE 1385

PRL_ERR_VM_BAD_OS_TYPE macro 1385

PRL_ERR_VM_CANT_CLONE_HDD_MISSING 1385

PRL_ERR_VM_CANT_CLONE_HDD_MISSING macro 1385

PRL_ERR_VM_CANT_CLONE_RUNNING 1385

PRL_ERR_VM_CANT_CLONE_RUNNING macro 1385

PRL_ERR_VM_CANT_DELETE_RUNNING 1385

PRL_ERR_VM_CANT_DELETE_RUNNING macro 1385

PRL_ERR_VM_CANT_UNREG_RUNNING 1385

PRL_ERR_VM_CANT_UNREG_RUNNING macro 1385

PRL_ERR_VM_COMPACT_HARD_DISK_FAILED 1386

PRL_ERR_VM_COMPACT_HARD_DISK_FAILED macro 1386

PRL_ERR_VM_COMPACT_NOT_SUPPORTED_GUEST_OS 1386

PRL_ERR_VM_COMPACT_NOT_SUPPORTED_GUEST_OS macro 1386

PRL_ERR_VM_COMPACT_PROCESSING 1386

PRL_ERR_VM_COMPACT_PROCESSING macro 1386

PRL_ERR_VM_CONF_CHANGED_INVALID_ARG 1386

PRL_ERR_VM_CONF_CHANGED_INVALID_ARG macro 1386

PRL_ERR_VM_CONF_CHANGED_NOT_CREATED 1387

PRL_ERR_VM_CONF_CHANGED_NOT_CREATED macro 1387

PRL_ERR_VM_CONF_CHANGED_NOT_STARTED 1387

PRL_ERR_VM_CONF_CHANGED_NOT_STARTED macro 1387

PRL_ERR_VM_CONF_CHANGED_UNSUPPORTED_DEV 1387

PRL_ERR_VM_CONF_CHANGED_UNSUPPORTED_DEV macro 1387

PRL_ERR_VM_CONFIG_ALREADY_EXISTS 1387

PRL_ERR_VM_CONFIG_ALREADY_EXISTS macro 1387

PRL_ERR_VM_CONFIG_CAN_BE_RESTORED 1387

PRL_ERR_VM_CONFIG_CAN_BE_RESTORED macro 1387

PRL_ERR_VM_CONFIG_DOESNT_EXIST 1388

PRL_ERR_VM_CONFIG_DOESNT_EXIST macro 1388

PRL_ERR_VM_CONFIG_INVALID_SERVER_UUID 1388

PRL_ERR_VM_CONFIG_INVALID_SERVER_UUID macro 1388

PRL_ERR_VM_CONFIG_INVALID_VM_UUID 1388

PRL_ERR_VM_CONFIG_INVALID_VM_UUID macro 1388

PRL_ERR_VM_CONFIG_IS_ALREADY_VALID 1388

PRL_ERR_VM_CONFIG_IS_ALREADY_VALID macro 1388

PRL_ERR_VM_CONFIG_WAS_CHANGED 1389

PRL_ERR_VM_CONFIG_WAS_CHANGED macro 1389

PRL_ERR_VM_COULDNT_BE_STARTED_UNDER_SPECIFIED_USER
1389

PRL_ERR_VM_COULDNT_BE_STARTED_UNDER_SPECIFIED_USER
macro 1389

PRL_ERR_VM_CREATE_HDD_IMG_INVALID_ARG 1389

PRL_ERR_VM_CREATE_HDD_IMG_INVALID_ARG macro 1389

PRL_ERR_VM_CREATE_HDD_IMG_INVALID_CREATE 1389

PRL_ERR_VM_CREATE_HDD_IMG_INVALID_CREATE macro 1389

PRL_ERR_VM_CREATE_INVALID_ARG 1390

PRL_ERR_VM_CREATE_INVALID_ARG macro 1390

PRL_ERR_VM_CREATE_SNAPSHOT_FAILED 1390

PRL_ERR_VM_CREATE_SNAPSHOT_FAILED macro 1390

PRL_ERR_VM_DELETE_STATE_FAILED 1390

PRL_ERR_VM_DELETE_STATE_FAILED macro 1390

PRL_ERR_VM_DEV_CHANGE_MEDIA_FAILED 1390

PRL_ERR_VM_DEV_CHANGE_MEDIA_FAILED macro 1390

PRL_ERR_VM_DEV_CONNECT_FAILED 1390

PRL_ERR_VM_DEV_CONNECT_FAILED macro 1390

PRL_ERR_VM_DEV_DISCONNECT_FAILED 1391

PRL_ERR_VM_DEV_DISCONNECT_FAILED macro 1391

PRL_ERR_VM_DEVICES_INITIALIZATION_FAILED 1391

PRL_ERR_VM_DEVICES_INITIALIZATION_FAILED macro 1391

PRL_ERR_VM_DEVICES_TERMINATION_FAILED 1391

PRL_ERR_VM_DEVICES_TERMINATION_FAILED macro 1391

PRL_ERR_VM_DIR_CONFIG_ALREADY_EXISTS 1391

PRL_ERR_VM_DIR_CONFIG_ALREADY_EXISTS macro 1391

PRL_ERR_VM_DIR_FILE_NOT_SET 1391

PRL_ERR_VM_DIR_FILE_NOT_SET macro 1391

PRL_ERR_VM_DIRECTORY_FOLDER_DOESNT_EXIST 1392

PRL_ERR_VM_DIRECTORY_FOLDER_DOESNT_EXIST macro 1392

PRL_ERR_VM_DIRECTORY_NOT_EXIST 1392

PRL_ERR_VM_DIRECTORY_NOT_EXIST macro 1392

PRL_ERR_VM_DIRECTORY_NOT_INITIALIZED 1392

PRL_ERR_VM_DIRECTORY_NOT_INITIALIZED macro 1392

PRL_ERR_VM_DOES_NOT_STOPPED 1392

PRL_ERR_VM_DOES_NOT_STOPPED macro 1392

PRL_ERR_VM_DVD_DISCONNECT_FAILED_LOCKED 1392

PRL_ERR_VM_DVD_DISCONNECT_FAILED_LOCKED macro 1392

PRL_ERR_VM_EDIT_UNABLE_CHANGE_IP_ADDRESS 1393

PRL_ERR_VM_EDIT_UNABLE_CHANGE_IP_ADDRESS macro 1393

PRL_ERR_VM_EDIT_UNABLE_CONVERT_TO_TEMPLATE 1393

PRL_ERR_VM_EDIT_UNABLE_CONVERT_TO_TEMPLATE macro 1393

PRL_ERR_VM_EDIT_UNABLE_SWITCH_OFF_UNDO_DISKS_MODE 1393

PRL_ERR_VM_EDIT_UNABLE_SWITCH_OFF_UNDO_DISKS_MODE macro 1393

PRL_ERR_VM_EDIT_UNABLE_SWITCH_ON_UNDO_DISKS_MODE 1393

PRL_ERR_VM_EDIT_UNABLE_SWITCH_ON_UNDO_DISKS_MODE macro 1393

PRL_ERR_VM_EMPTY_NAME_OF_CLONE 1394

PRL_ERR_VM_EMPTY_NAME_OF_CLONE macro 1394

PRL_ERR_VM_EXEC_GUEST_TOOL_NOT_AVAILABLE 1394

PRL_ERR_VM_EXEC_GUEST_TOOL_NOT_AVAILABLE macro 1394

PRL_ERR_VM_EXEC_PROGRAM_NOT_FOUND 1394

PRL_ERR_VM_EXEC_PROGRAM_NOT_FOUND macro 1394

PRL_ERR_VM_EXPIRATION_HOST_DATETIME_SKEWED 1394

PRL_ERR_VM_EXPIRATION_HOST_DATETIME_SKEWED macro 1394

PRL_ERR_VM_EXPIRATION_OFFLINE_GRACE_PERIOD_EXPIRED 1395

PRL_ERR_VM_EXPIRATION_OFFLINE_GRACE_PERIOD_EXPIRED macro 1395

PRL_ERR_VM_EXPIRATION_TIME_CHECK_INTERVAL_OUT_OF_RANGE 1395

PRL_ERR_VM_EXPIRATION_TIME_CHECK_INTERVAL_OUT_OF_RANGE macro 1395

PRL_ERR_VM_EXPIRATION_VM_IS_ABOUT_TO_EXPIRE 1395

PRL_ERR_VM_EXPIRATION_VM_IS_ABOUT_TO_EXPIRE macro 1395

PRL_ERR_VM_EXPIRATION_VM_IS_IN_OFFLINE_GRACE_PERIOD 1395

PRL_ERR_VM_EXPIRATION_VM_IS_IN_OFFLINE_GRACE_PERIOD macro 1395

PRL_ERR_VM_FILES_ALREADY_REMOVED 1396

PRL_ERR_VM_FILES_ALREADY_REMOVED macro 1396
PRL_ERR_VM_FREE_VM_MEMORY 1396
PRL_ERR_VM_FREE_VM_MEMORY macro 1396
PRL_ERR_VM_GET_CONFIG_FAILED 1396
PRL_ERR_VM_GET_CONFIG_FAILED macro 1396
PRL_ERR_VM_GET_CONFIG_INVALID_ARG 1396
PRL_ERR_VM_GET_CONFIG_INVALID_ARG macro 1396
PRL_ERR_VM_GET_CONFIG_NOT_CONFIGURED 1397
PRL_ERR_VM_GET_CONFIG_NOT_CONFIGURED macro 1397
PRL_ERR_VM_GET_DEVICE_STATE_FAILED 1397
PRL_ERR_VM_GET_DEVICE_STATE_FAILED macro 1397
PRL_ERR_VM_GET_HDD_IMG_INVALID_ARG 1397
PRL_ERR_VM_GET_HDD_IMG_INVALID_ARG macro 1397
PRL_ERR_VM_GET_HDD_IMG_NOT_OPEN 1397
PRL_ERR_VM_GET_HDD_IMG_NOT_OPEN macro 1397
PRL_ERR_VM_GET_INDICATORS_INVALID_ARG 1397
PRL_ERR_VM_GET_INDICATORS_INVALID_ARG macro 1397
PRL_ERR_VM_GET_INDICATORS_NOT_CREATED 1398
PRL_ERR_VM_GET_INDICATORS_NOT_CREATED macro 1398
PRL_ERR_VM_GET_INDICATORS_NOT_RUNNING 1398
PRL_ERR_VM_GET_INDICATORS_NOT_RUNNING macro 1398
PRL_ERR_VM_GET_PROBLEM_REPORT_FAILED 1398
PRL_ERR_VM_GET_PROBLEM_REPORT_FAILED macro 1398
PRL_ERR_VM_GET_SCR_SIZE_NOT_CREATED 1398
PRL_ERR_VM_GET_SCR_SIZE_NOT_CREATED macro 1398
PRL_ERR_VM_GET_SCR_UPDATED_NOT_CREATED 1398
PRL_ERR_VM_GET_SCR_UPDATED_NOT_CREATED macro 1398
PRL_ERR_VM_GET_SCR_UPDATED_NOT_RUNNING 1399
PRL_ERR_VM_GET_SCR_UPDATED_NOT_RUNNING macro 1399
PRL_ERR_VM_GET_STATUS_FAILED 1399
PRL_ERR_VM_GET_STATUS_FAILED macro 1399
PRL_ERR_VM_GET_STATUS_INVALID_ARG 1399
PRL_ERR_VM_GET_STATUS_INVALID_ARG macro 1399
PRL_ERR_VM_GUEST_SESSION_EXPIRED 1399
PRL_ERR_VM_GUEST_SESSION_EXPIRED macro 1399
PRL_ERR_VM_GUESTMEM_FAIL 1399
PRL_ERR_VM_GUESTMEM_FAIL macro 1399
PRL_ERR_VM_HARD_DISK_NOT_COMPACTABLE 1400
PRL_ERR_VM_HARD_DISK_NOT_COMPACTABLE macro 1400
PRL_ERR_VM_HDD_DISCONNECT_FAILED_LOCKED 1400
PRL_ERR_VM_HDD_DISCONNECT_FAILED_LOCKED macro 1400
PRL_ERR_VM_HDD_SIZE 1400
PRL_ERR_VM_HDD_SIZE macro 1400
PRL_ERR_VM_HOME_PATH_IS_NOT_EMPTY 1400
PRL_ERR_VM_HOME_PATH_IS_NOT_EMPTY macro 1400
PRL_ERR_VM_HYPERVISOR_HVT_ENABLED_ON_HYPERSWITCH 1401
PRL_ERR_VM_HYPERVISOR_HVT_ENABLED_ON_HYPERSWITCH macro 1401
PRL_ERR_VM_HYPERVIZOR_COMM 1401
PRL_ERR_VM_HYPERVIZOR_COMM macro 1401
PRL_ERR_VM_IN_FROZEN_STATE 1401
PRL_ERR_VM_IN_FROZEN_STATE macro 1401
PRL_ERR_VM_IN_HDD_CONSTRUCTOR 1401
PRL_ERR_VM_IN_HDD_CONSTRUCTOR macro 1401
PRL_ERR_VM_IN_WAKING_UP_STATE 1402
PRL_ERR_VM_IN_WAKING_UP_STATE macro 1402
PRL_ERR_VM_INIT_MONITOR 1402
PRL_ERR_VM_INIT_MONITOR macro 1402
PRL_ERR_VM_INIT_VCPU 1402
PRL_ERR_VM_INIT_VCPU macro 1402
PRL_ERR_VM_INTERACT_PRLS_DRIVER 1402
PRL_ERR_VM_INTERACT_PRLS_DRIVER macro 1402
PRL_ERR_VM_INTERNAL_OS_ERROR 1402
PRL_ERR_VM_INTERNAL_OS_ERROR macro 1402
PRL_ERR_VM_INVALID_SWAP_REGION 1403
PRL_ERR_VM_INVALID_SWAP_REGION macro 1403
PRL_ERR_VM_IS_EXCLUSIVELY_LOCKED 1403
PRL_ERR_VM_IS_EXCLUSIVELY_LOCKED macro 1403
PRL_ERR_VM_IS_NOT_LOCKED 1403
PRL_ERR_VM_IS_NOT_LOCKED macro 1403
PRL_ERR_VM_IS_NOT_SUSPENDED 1403

PRL_ERR_VM_IS_NOT_SUSPENDED macro 1403
PRL_ERR_VM_LIFETIME_IS_EXPIRED 1403
PRL_ERR_VM_LIFETIME_IS_EXPIRED macro 1403
PRL_ERR_VM_LOAD_BINARY_FAILED 1404
PRL_ERR_VM_LOAD_BINARY_FAILED macro 1404
PRL_ERR_VM_LOAD_MONITOR 1404
PRL_ERR_VM_LOAD_MONITOR macro 1404
PRL_ERR_VM_LOCKED_CTL_FOR_BACKUP 1404
PRL_ERR_VM_LOCKED_CTL_FOR_BACKUP macro 1404
PRL_ERR_VM_LOCKED_FOR_BACKUP 1404
PRL_ERR_VM_LOCKED_FOR_BACKUP macro 1404
PRL_ERR_VM_LOCKED_FOR_CHANGE_FIREWALL 1404
PRL_ERR_VM_LOCKED_FOR_CHANGE_FIREWALL macro 1404
PRL_ERR_VM_LOCKED_FOR_CHANGE_PASSWORD 1405
PRL_ERR_VM_LOCKED_FOR_CHANGE_PASSWORD macro 1405
PRL_ERR_VM_LOCKED_FOR_CLONE 1405
PRL_ERR_VM_LOCKED_FOR_CLONE macro 1405
PRL_ERR_VM_LOCKED_FOR_COPY_IMAGE 1405
PRL_ERR_VM_LOCKED_FOR_COPY_IMAGE macro 1405
PRL_ERR_VM_LOCKED_FOR_CREATE_SNAPSHOT 1405
PRL_ERR_VM_LOCKED_FOR_CREATE_SNAPSHOT macro 1405
PRL_ERR_VM_LOCKED_FOR_DECRYPT 1405
PRL_ERR_VM_LOCKED_FOR_DECRYPT macro 1405
PRL_ERR_VM_LOCKED_FOR_DELETE 1406
PRL_ERR_VM_LOCKED_FOR_DELETE macro 1406
PRL_ERR_VM_LOCKED_FOR_DELETE_TO_SNAPSHOT 1406
PRL_ERR_VM_LOCKED_FOR_DELETE_TO_SNAPSHOT macro 1406
PRL_ERR_VM_LOCKED_FOR_DISK_COMPACT 1406
PRL_ERR_VM_LOCKED_FOR_DISK_COMPACT macro 1406
PRL_ERR_VM_LOCKED_FOR_DISK_CONVERT 1406
PRL_ERR_VM_LOCKED_FOR_DISK_CONVERT macro 1406
PRL_ERR_VM_LOCKED_FOR_DISK_RESIZE 1407
PRL_ERR_VM_LOCKED_FOR_DISK_RESIZE macro 1407
PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT 1407
PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT macro 1407
PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT_WITH_RENAME 1407
PRL_ERR_VM_LOCKED_FOR_EDIT_COMMIT_WITH_RENAME macro 1407
PRL_ERR_VM_LOCKED_FOR_ENCRYPT 1407
PRL_ERR_VM_LOCKED_FOR_ENCRYPT macro 1407
PRL_ERR_VM_LOCKED_FOR_EXECUTE 1408
PRL_ERR_VM_LOCKED_FOR_EXECUTE macro 1408
PRL_ERR_VM_LOCKED_FOR_EXECUTE_EX 1408
PRL_ERR_VM_LOCKED_FOR_EXECUTE_EX macro 1408
PRL_ERR_VM_LOCKED_FOR_INTERNAL_REASON 1408
PRL_ERR_VM_LOCKED_FOR_INTERNAL_REASON macro 1408
PRL_ERR_VM_LOCKED_FOR_MIGRATE 1408
PRL_ERR_VM_LOCKED_FOR_MIGRATE macro 1408
PRL_ERR_VM_LOCKED_FOR_MOVE 1408
PRL_ERR_VM_LOCKED_FOR_MOVE macro 1408
PRL_ERR_VM_LOCKED_FOR_REMOVE_PROTECTION 1409
PRL_ERR_VM_LOCKED_FOR_REMOVE_PROTECTION macro 1409
PRL_ERR_VM_LOCKED_FOR_RESTORE_FROM_BACKUP 1409
PRL_ERR_VM_LOCKED_FOR_RESTORE_FROM_BACKUP macro 1409
PRL_ERR_VM_LOCKED_FOR_SET_PROTECTION 1409
PRL_ERR_VM_LOCKED_FOR_SET_PROTECTION macro 1409
PRL_ERR_VM_LOCKED_FOR_SWITCH_TO_SNAPSHOT 1409
PRL_ERR_VM_LOCKED_FOR_SWITCH_TO_SNAPSHOT macro 1409
PRL_ERR_VM_LOCKED_FOR_UNREGISTER 1410
PRL_ERR_VM_LOCKED_FOR_UNREGISTER macro 1410
PRL_ERR_VM_LOCKED_FOR_UPDATE_SECURITY 1410
PRL_ERR_VM_LOCKED_FOR_UPDATE_SECURITY macro 1410
PRL_ERR_VM_MAP_VM_MEM0 1410
PRL_ERR_VM_MAP_VM_MEM0 macro 1410
PRL_ERR_VM_MEMORY_SWAPPING_IN_PROGRESS 1410
PRL_ERR_VM_MEMORY_SWAPPING_IN_PROGRESS macro 1410
PRL_ERR_VM_MIGRATE_ACCESS_TO_VM_DENIED 1410
PRL_ERR_VM_MIGRATE_ACCESS_TO_VM_DENIED macro

1410

PRL_ERR_VM_MIGRATE_BREAK_BY_DISK_CONDITION 1411

PRL_ERR_VM_MIGRATE_BREAK_BY_DISK_CONDITION macro 1411

PRL_ERR_VM_MIGRATE_CANNOT_CREATE_DIRECTORY 1411

PRL_ERR_VM_MIGRATE_CANNOT_CREATE_DIRECTORY macro 1411

PRL_ERR_VM_MIGRATE_CANNOT_REMOTE_CLONE_SHARED_VM 1411

PRL_ERR_VM_MIGRATE_CANNOT_REMOTE_CLONE_SHARED_VM macro 1411

PRL_ERR_VM_MIGRATE_CHECKING_PRECONDITIONS_FAILED 1411

PRL_ERR_VM_MIGRATE_CHECKING_PRECONDITIONS_FAILED macro 1411

PRL_ERR_VM_MIGRATE_CONTINUE_START_FAILED 1412

PRL_ERR_VM_MIGRATE_CONTINUE_START_FAILED macro 1412

PRL_ERR_VM_MIGRATE_COULDNT_DETACH_TARGET_CONNECTION 1412

PRL_ERR_VM_MIGRATE_COULDNT_DETACH_TARGET_CONNECTION macro 1412

PRL_ERR_VM_MIGRATE_DEVICE_IMAGE_OUT_OF_BUNDLE 1412

PRL_ERR_VM_MIGRATE_DEVICE_IMAGE_OUT_OF_BUNDLE macro 1412

PRL_ERR_VM_MIGRATE_ERROR_DELETE_VM 1412

PRL_ERR_VM_MIGRATE_ERROR_DELETE_VM macro 1412

PRL_ERR_VM_MIGRATE_ERROR_UNREGISTER_VM 1413

PRL_ERR_VM_MIGRATE_ERROR_UNREGISTER_VM macro 1413

PRL_ERR_VM_MIGRATE_EXT_DISK_DIR_ALREADY_EXISTS_ON_TARGET 1413

PRL_ERR_VM_MIGRATE_EXT_DISK_DIR_ALREADY_EXISTS_ON_TARGET macro 1413

PRL_ERR_VM_MIGRATE_EXTERNAL_DISKS_NOT_SUPPORTED 1413

PRL_ERR_VM_MIGRATE_EXTERNAL_DISKS_NOT_SUPPORTED macro 1413

PRL_ERR_VM_MIGRATE_FLOPPY_DISK_IS_ABSENT_ON_TARGET 1413

PRL_ERR_VM_MIGRATE_FLOPPY_DISK_IS_ABSENT_ON_TARGET macro 1413

PRL_ERR_VM_MIGRATE_INVALID_DISK_TYPE 1414

PRL_ERR_VM_MIGRATE_INVALID_DISK_TYPE macro 1414

PRL_ERR_VM_MIGRATE_NETWORK_ADAPTER_IS_ABSENT_ON_TARGET 1414

PRL_ERR_VM_MIGRATE_NETWORK_ADAPTER_IS_ABSENT_ON_TARGET macro 1414

PRL_ERR_VM_MIGRATE_NETWORK_SHARE_IS_ABSENT_ON_TARGET 1414

PRL_ERR_VM_MIGRATE_NETWORK_SHARE_IS_ABSENT_ON_TARGET macro 1414

PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET 1415

PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET macro 1415

PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET_SHORT 1415

PRL_ERR_VM_MIGRATE_NON_COMPATIBLE_CPU_ON_TARGET_SHORT macro 1415

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_CPUS_ON_TARGET 1415

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_CPUS_ON_TARGET macro 1415

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_SOURCE 1416

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_SOURCE macro 1416

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_TARGET 1416

PRL_ERR_VM_MIGRATE_NOT_ENOUGH_DISK_SPACE_ON_TARGET
macro 1416

PRL_ERR_VM_MIGRATE_OPTICAL_DISK_IS_ABSENT_ON_TARGET
1416

PRL_ERR_VM_MIGRATE_OPTICAL_DISK_IS_ABSENT_ON_TARGET
macro 1416

PRL_ERR_VM_MIGRATE_OUT_OF_MEMORY_ON_TARGET
1416

PRL_ERR_VM_MIGRATE_OUT_OF_MEMORY_ON_TARGET
macro 1416

PRL_ERR_VM_MIGRATE_PARALLEL_PORT_IS_ABSENT_ON_TARGET
1417

PRL_ERR_VM_MIGRATE_PARALLEL_PORT_IS_ABSENT_ON_TARGET
macro 1417

PRL_ERR_VM_MIGRATE_REGISTER_VM_FAILED 1417

PRL_ERR_VM_MIGRATE_REGISTER_VM_FAILED macro 1417

PRL_ERR_VM_MIGRATE_REMOTE_DEVICE_IS_ATTACHED 1417

PRL_ERR_VM_MIGRATE_REMOTE_DEVICE_IS_ATTACHED macro 1417

PRL_ERR_VM_MIGRATE_RESUME_FAILED 1417

PRL_ERR_VM_MIGRATE_RESUME_FAILED macro 1417

PRL_ERR_VM_MIGRATE_RESUME_VM_FAILED 1418

PRL_ERR_VM_MIGRATE_RESUME_VM_FAILED macro 1418

PRL_ERR_VM_MIGRATE_SERIAL_PORT_IS_ABSENT_ON_TARGET
1418

PRL_ERR_VM_MIGRATE_SERIAL_PORT_IS_ABSENT_ON_TARGET
macro 1418

PRL_ERR_VM_MIGRATE_SOUND_DEVICE_IS_ABSENT_ON_TARGET
1418

PRL_ERR_VM_MIGRATE_SOUND_DEVICE_IS_ABSENT_ON_TARGET
macro 1418

PRL_ERR_VM_MIGRATE_STORAGE_INFO_PARSE 1418

PRL_ERR_VM_MIGRATE_STORAGE_INFO_PARSE macro 1418

PRL_ERR_VM_MIGRATE_SUSPEND_VM_FAILED 1419

PRL_ERR_VM_MIGRATE_SUSPEND_VM_FAILED macro 1419

PRL_ERR_VM_MIGRATE_TARGET_INSIDE_SHARED_VM_PRIVATE
1419

PRL_ERR_VM_MIGRATE_TARGET_INSIDE_SHARED_VM_PRIVATE
macro 1419

PRL_ERR_VM_MIGRATE_TARGET_VM_HOME_PATH_NOT_EXISTS
1419

PRL_ERR_VM_MIGRATE_TARGET_VM_HOME_PATH_NOT_EXISTS
macro 1419

PRL_ERR_VM_MIGRATE_TO_THE_SAME_NODE 1419

PRL_ERR_VM_MIGRATE_TO_THE_SAME_NODE macro 1419

PRL_ERR_VM_MIGRATE_UNSUITABLE_VM_STATE 1420

PRL_ERR_VM_MIGRATE_UNSUITABLE_VM_STATE macro 1420

PRL_ERR_VM_MIGRATE_USB_CONTROLLER_IS_ABSENT_ON_TARGET
1420

PRL_ERR_VM_MIGRATE_USB_CONTROLLER_IS_ABSENT_ON_TARGET
macro 1420

PRL_ERR_VM_MIGRATE_VM_ALREADY_EXISTS_ON_TARGET
1420

PRL_ERR_VM_MIGRATE_VM_ALREADY_EXISTS_ON_TARGET
macro 1420

PRL_ERR_VM_MIGRATE_VM_ALREADY_MIGRATE_ON_TARGET
1420

PRL_ERR_VM_MIGRATE_VM_ALREADY_MIGRATE_ON_TARGET
macro 1420

PRL_ERR_VM_MIGRATE_VM_HOME_ALREADY_EXISTS_ON_TARGET
1421

PRL_ERR_VM_MIGRATE_VM_HOME_ALREADY_EXISTS_ON_TARGET
macro 1421

PRL_ERR_VM_MIGRATE_VM_UUID_ALREADY_EXISTS_ON_TARGET
1421

PRL_ERR_VM_MIGRATE_VM_UUID_ALREADY_EXISTS_ON_TARGET
macro 1421

PRL_ERR_VM_MIGRATE_WARM_MODE_NOT_SUPPORTED

1421

PRL_ERR_VM_MIGRATE_WARM_MODE_NOT_SUPPORTED macro 1421
PRL_ERR_VM_MONITOR 1421
PRL_ERR_VM_MONITOR macro 1421
PRL_ERR_VM_MOUNT 1422
PRL_ERR_VM_MOUNT macro 1422
PRL_ERR_VM_MUST_BE_STOPPED_BEFORE_RENAMING 1422
PRL_ERR_VM_MUST_BE_STOPPED_BEFORE_RENAMING macro 1422
PRL_ERR_VM_MUST_BE_STOPPED_FOR_CHANGE_DEVICES 1422
PRL_ERR_VM_MUST_BE_STOPPED_FOR_CHANGE_DEVICES macro 1422
PRL_ERR_VM_NAME_IS_EMPTY 1422
PRL_ERR_VM_NAME_IS_EMPTY macro 1422
PRL_ERR_VM_NOT_CREATED 1423
PRL_ERR_VM_NOT_CREATED macro 1423
PRL_ERR_VM_OPERATION_FAILED 1423
PRL_ERR_VM_OPERATION_FAILED macro 1423
PRL_ERR_VM_PAUSE_ALREADY_PAUSED 1423
PRL_ERR_VM_PAUSE_ALREADY_PAUSED macro 1423
PRL_ERR_VM_PAUSE_FAILED 1423
PRL_ERR_VM_PAUSE_FAILED macro 1423
PRL_ERR_VM_PAUSE_NOT_CREATED 1423
PRL_ERR_VM_PAUSE_NOT_CREATED macro 1423
PRL_ERR_VM_PAUSE_NOT_STARTED 1424
PRL_ERR_VM_PAUSE_NOT_STARTED macro 1424
PRL_ERR_VM_POWER_OFF_FAILED 1424
PRL_ERR_VM_POWER_OFF_FAILED macro 1424
PRL_ERR_VM_POWER_ON_FAILED 1424
PRL_ERR_VM_POWER_ON_FAILED macro 1424
PRL_ERR_VM_PRLS_DESCTOP_ALLOC_PHYS_MEM 1424
PRL_ERR_VM_PRLS_DESCTOP_ALLOC_PHYS_MEM macro 1424
PRL_ERR_VM_PROCESS_IS_NOT_STARTED 1424
PRL_ERR_VM_PROCESS_IS_NOT_STARTED macro 1424
PRL_ERR_VM_PROCESS_IS_NOT_STARTED_BY_WRONG_HOST_SW 1425
PRL_ERR_VM_PROCESS_IS_NOT_STARTED_BY_WRONG_HOST_SW macro 1425
PRL_ERR_VM_REQUEST_NOT_SUPPORTED 1425
PRL_ERR_VM_REQUEST_NOT_SUPPORTED macro 1425
PRL_ERR_VM_RESET_FAILED 1425
PRL_ERR_VM_RESET_FAILED macro 1425
PRL_ERR_VM_RESTART_GUEST_FAILED 1425
PRL_ERR_VM_RESTART_GUEST_FAILED macro 1425
PRL_ERR_VM_RESTART_NOT_CREATED 1426
PRL_ERR_VM_RESTART_NOT_CREATED macro 1426
PRL_ERR_VM_RESTART_NOT_STARTED 1426
PRL_ERR_VM_RESTART_NOT_STARTED macro 1426
PRL_ERR_VM_RESTORE_STATE_FAILED 1426
PRL_ERR_VM_RESTORE_STATE_FAILED macro 1426
PRL_ERR_VM_RESTORE_STATE_FAILED_VM_STOPPED 1426
PRL_ERR_VM_RESTORE_STATE_FAILED_VM_STOPPED macro 1426
PRL_ERR_VM_RESUME_FAILED 1426
PRL_ERR_VM_RESUME_FAILED macro 1426
PRL_ERR_VM_RESUME_INV_SAV_VERSION_CANCEL_RESUME 1427
PRL_ERR_VM_RESUME_INV_SAV_VERSION_CANCEL_RESUME macro 1427
PRL_ERR_VM_SAVE_STATE_FAILED 1427
PRL_ERR_VM_SAVE_STATE_FAILED macro 1427
PRL_ERR_VM_SEND_ABS_MOUSE_NOT_CREATED 1427
PRL_ERR_VM_SEND_ABS_MOUSE_NOT_CREATED macro 1427
PRL_ERR_VM_SEND_ABS_MOUSE_NOT_RUNNING 1427
PRL_ERR_VM_SEND_ABS_MOUSE_NOT_RUNNING macro 1427
PRL_ERR_VM_SEND_ABS_MOUSE_PAUSED 1428
PRL_ERR_VM_SEND_ABS_MOUSE_PAUSED macro 1428
PRL_ERR_VM_SEND_KEYBOARD_NOT_CREATED 1428
PRL_ERR_VM_SEND_KEYBOARD_NOT_CREATED macro 1428
PRL_ERR_VM_SEND_KEYBOARD_NOT_RUNNING 1428
PRL_ERR_VM_SEND_KEYBOARD_NOT_RUNNING macro 1428
PRL_ERR_VM_SEND_KEYBOARD_PAUSED 1428

PRL_ERR_VM_SEND_KEYBOARD_PAUSED macro 1428
PRL_ERR_VM_SEND_REL_MOUSE_NOT_CREATED 1428
PRL_ERR_VM_SEND_REL_MOUSE_NOT_CREATED macro 1428
PRL_ERR_VM_SEND_REL_MOUSE_NOT_RUNNING 1429
PRL_ERR_VM_SEND_REL_MOUSE_NOT_RUNNING macro 1429
PRL_ERR_VM_SEND_REL_MOUSE_PAUSED 1429
PRL_ERR_VM_SEND_REL_MOUSE_PAUSED macro 1429
PRL_ERR_VM_SET_CONFIG_FAILED 1429
PRL_ERR_VM_SET_CONFIG_FAILED macro 1429
PRL_ERR_VM_SET_CONFIG_INVALID_ARG 1429
PRL_ERR_VM_SET_CONFIG_INVALID_ARG macro 1429
PRL_ERR_VM_SET_LICENSE_FAILED 1429
PRL_ERR_VM_SET_LICENSE_FAILED macro 1429
PRL_ERR_VM_SET_VISIBLE_NOT_CREATED 1430
PRL_ERR_VM_SET_VISIBLE_NOT_CREATED macro 1430
PRL_ERR_VM_SHUTDOWN_FAILED 1430
PRL_ERR_VM_SHUTDOWN_FAILED macro 1430
PRL_ERR_VM_SHUTDOWN_HIBERNATE_FAILED 1430
PRL_ERR_VM_SHUTDOWN_HIBERNATE_FAILED macro 1430
PRL_ERR_VM_SHUTDOWN_HIBERNATE_NOT_SUPPORTED 1430
PRL_ERR_VM_SHUTDOWN_HIBERNATE_NOT_SUPPORTED macro 1430
PRL_ERR_VM_SHUTDOWN_MACHINE_LOCKED 1431
PRL_ERR_VM_SHUTDOWN_MACHINE_LOCKED macro 1431
PRL_ERR_VM_SHUTDOWN_SUSPEND_NOT_SUPPORTED 1431
PRL_ERR_VM_SHUTDOWN_SUSPEND_NOT_SUPPORTED macro 1431
PRL_ERR_VM_SNAPSHOT_CHANGED_VM_CONFIG 1431
PRL_ERR_VM_SNAPSHOT_CHANGED_VM_CONFIG macro 1431
PRL_ERR_VM_SNAPSHOT_IN_SAFE_MODE 1431
PRL_ERR_VM_SNAPSHOT_IN_SAFE_MODE macro 1431
PRL_ERR_VM_SNAPSHOT_IN_UNDO_DISKS_MODE 1431
PRL_ERR_VM_SNAPSHOT_IN_UNDO_DISKS_MODE macro 1431
PRL_ERR_VM_SNAPSHOT_NOT_FOUND 1432
PRL_ERR_VM_SNAPSHOT_NOT_FOUND macro 1432
PRL_ERR_VM_SNAPSHOTS_CONFIG_NOT_FOUND 1432
PRL_ERR_VM_SNAPSHOTS_CONFIG_NOT_FOUND macro 1432
PRL_ERR_VM_SPECIFY_GUEST_INSTALL_FILES 1432
PRL_ERR_VM_SPECIFY_GUEST_INSTALL_FILES macro 1432
PRL_ERR_VM_START_FAILED 1432
PRL_ERR_VM_START_FAILED macro 1432
PRL_ERR_VM_START_NOT_CONFIGURED 1433
PRL_ERR_VM_START_NOT_CONFIGURED macro 1433
PRL_ERR_VM_START_NOT_CREATED 1433
PRL_ERR_VM_START_NOT_CREATED macro 1433
PRL_ERR_VM_STOP_NOT_CREATED 1433
PRL_ERR_VM_STOP_NOT_CREATED macro 1433
PRL_ERR_VM_SUSPEND_CHANGED_VM_CONFIG 1433
PRL_ERR_VM_SUSPEND_CHANGED_VM_CONFIG macro 1433
PRL_ERR_VM_SUSPEND_FAILED 1433
PRL_ERR_VM_SUSPEND_FAILED macro 1433
PRL_ERR_VM_TO_REGISTER_IS_NOT_SPECIFIED 1434
PRL_ERR_VM_TO_REGISTER_IS_NOT_SPECIFIED macro 1434
PRL_ERR_VM_TOOL_NOT_AVAILABLE 1434
PRL_ERR_VM_TOOL_NOT_AVAILABLE macro 1434
PRL_ERR_VM_TOOLS_CANT_PARSE_UPDATE_PARAMETERS 1434
PRL_ERR_VM_TOOLS_CANT_PARSE_UPDATE_PARAMETERS macro 1434
PRL_ERR_VM_TOOLS_CANT_UPDATE_WITHOUT_RESTART 1434
PRL_ERR_VM_TOOLS_CANT_UPDATE_WITHOUT_RESTART macro 1434
PRL_ERR_VM_UNABLE_ALLOC_MEM 1435
PRL_ERR_VM_UNABLE_ALLOC_MEM macro 1435
PRL_ERR_VM_UNABLE_ALLOC_MEM_MONITOR 1435
PRL_ERR_VM_UNABLE_ALLOC_MEM_MONITOR macro 1435
PRL_ERR_VM_UNABLE_CREATE_TIMER 1435
PRL_ERR_VM_UNABLE_CREATE_TIMER macro 1435

PRL_ERR_VM_UNABLE_GET_GUEST_CPU 1435

PRL_ERR_VM_UNABLE_GET_GUEST_CPU macro 1435

PRL_ERR_VM_UNABLE_OPEN_DISK_IMAGE 1435

PRL_ERR_VM_UNABLE_OPEN_DISK_IMAGE macro 1435

PRL_ERR_VM_UNABLE_SEND_REQUEST 1436

PRL_ERR_VM_UNABLE_SEND_REQUEST macro 1436

PRL_ERR_VM_UNABLE_TO_OPEN_VIRTUAL_BRIDGE 1436

PRL_ERR_VM_UNABLE_TO_OPEN_VIRTUAL_BRIDGE macro 1436

PRL_ERR_VM_UNDEFINED_API_CALL 1436

PRL_ERR_VM_UNDEFINED_API_CALL macro 1436

PRL_ERR_VM_UNMOUNT 1436

PRL_ERR_VM_UNMOUNT macro 1436

PRL_ERR_VM_UNPAUSE_FAILED 1436

PRL_ERR_VM_UNPAUSE_FAILED macro 1436

PRL_ERR_VM_UPDATE_SNAPSHOT_DATA_FAILED 1437

PRL_ERR_VM_UPDATE_SNAPSHOT_DATA_FAILED macro 1437

PRL_ERR_VM_USER_AUTHENTICATION_FAILED 1437

PRL_ERR_VM_USER_AUTHENTICATION_FAILED macro 1437

PRL_ERR_VM_UUID_EMPTY 1437

PRL_ERR_VM_UUID_EMPTY macro 1437

PRL_ERR_VM_UUID_NOT_FOUND 1437

PRL_ERR_VM_UUID_NOT_FOUND macro 1437

PRL_ERR_VM_VALLOC_MON_BODY 1438

PRL_ERR_VM_VALLOC_MON_BODY macro 1438

PRL_ERR_VM_VALLOC_PE_IMG 1438

PRL_ERR_VM_VALLOC_PE_IMG macro 1438

PRL_ERR_VM_VIEW_SCR_INVALID_ARG 1438

PRL_ERR_VM_VIEW_SCR_INVALID_ARG macro 1438

PRL_ERR_VM_VIEW_SCR_NOT_CREATED 1438

PRL_ERR_VM_VIEW_SCR_NOT_CREATED macro 1438

PRL_ERR_VM_VIEW_SCR_NOT_RUNNING 1438

PRL_ERR_VM_VIEW_SCR_NOT_RUNNING macro 1438

PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN
1439

PRL_ERR_VMCONF_AUTOSTART_FROM_CURRENT_USER_FORBIDDEN
macro 1439

PRL_ERR_VMCONF_BOOT_OPTION_DEVICE_NOT_EXISTS
1439

PRL_ERR_VMCONF_BOOT_OPTION_DEVICE_NOT_EXISTS
macro 1439

PRL_ERR_VMCONF_BOOT_OPTION_DUPLICATE_DEVICE
1439

PRL_ERR_VMCONF_BOOT_OPTION_DUPLICATE_DEVICE
macro 1439

PRL_ERR_VMCONF_BOOT_OPTION_INVALID_DEVICE_TYPE
1439

PRL_ERR_VMCONF_BOOT_OPTION_INVALID_DEVICE_TYPE
macro 1439

PRL_ERR_VMCONF_BOOTCAMP_HARD_DISK_SMART_GUARD_NOT_ALLOW
1440

PRL_ERR_VMCONF_BOOTCAMP_HARD_DISK_SMART_GUARD_NOT_ALLOW
macro 1440

PRL_ERR_VMCONF_BOOTCAMP_HARD_SNAPSHOTS_NOT_ALLOW
1440

PRL_ERR_VMCONF_BOOTCAMP_HARD_SNAPSHOTS_NOT_ALLOW
macro 1440

PRL_ERR_VMCONF_BOOTCAMP_HARD_UNDO_DISKS_NOT_ALLOW
1440

PRL_ERR_VMCONF_BOOTCAMP_HARD_UNDO_DISKS_NOT_ALLOW
macro 1440

PRL_ERR_VMCONF_BOOTCAMP_SAFE_MODE_NOT_ALLOW
1441

PRL_ERR_VMCONF_BOOTCAMP_SAFE_MODE_NOT_ALLOW
macro 1441

PRL_ERR_VMCONF_CD_DVD_ROM_DUPLICATE_SYS_NAME
1441

PRL_ERR_VMCONF_CD_DVD_ROM_DUPLICATE_SYS_NAME
macro 1441

PRL_ERR_VMCONF_CD_DVD_ROM_IMAGE_IS_NOT_EXIST
1441

PRL_ERR_VMCONF_CD_DVD_ROM_IMAGE_IS_NOT_EXIST
macro 1441

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_OLD_CHIPSET

1441

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_OLD_CHIPSET
macro 1441

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_UNSUPPORTED_OS
1442

PRL_ERR_VMCONF_CD_DVD_ROM_SET_SATA_FOR_UNSUPPORTED_OS
macro 1442

PRL_ERR_VMCONF_CD_DVD_ROM_SYS_NAME_IS_EMPTY 1442

PRL_ERR_VMCONF_CD_DVD_ROM_SYS_NAME_IS_EMPTY macro 1442

PRL_ERR_VMCONF_CD_DVD_ROM_URL_FORMAT_SYS_NAME
1442

PRL_ERR_VMCONF_CD_DVD_ROM_URL_FORMAT_SYS_NAME
macro 1442

PRL_ERR_VMCONF_CPU_COUNT_MORE_HOST_CPU_COUNT
1443

PRL_ERR_VMCONF_CPU_COUNT_MORE_HOST_CPU_COUNT
macro 1443

PRL_ERR_VMCONF_CPU_COUNT_MORE_MAX_CPU_COUNT
1443

PRL_ERR_VMCONF_CPU_COUNT_MORE_MAX_CPU_COUNT
macro 1443

PRL_ERR_VMCONF_CPU_MASK_INVALID 1443

PRL_ERR_VMCONF_CPU_MASK_INVALID macro 1443

PRL_ERR_VMCONF_CPU_MASK_INVALID_CPU_NUM 1443

PRL_ERR_VMCONF_CPU_MASK_INVALID_CPU_NUM macro 1443

PRL_ERR_VMCONF_CPU_NODE_MASK_INVALID 1444

PRL_ERR_VMCONF_CPU_NODE_MASK_INVALID macro 1444

PRL_ERR_VMCONF_CPU_ZERO_COUNT 1444

PRL_ERR_VMCONF_CPU_ZERO_COUNT macro 1444

PRL_ERR_VMCONF_CPULIMIT_NOT_SUPPORTED 1444

PRL_ERR_VMCONF_CPULIMIT_NOT_SUPPORTED macro 1444

PRL_ERR_VMCONF_CPUUNITS_NOT_SUPPORTED 1444

PRL_ERR_VMCONF_CPUUNITS_NOT_SUPPORTED macro 1444

PRL_ERR_VMCONF_DESKTOP_MODE_REMOTE_DEVICES 1445

PRL_ERR_VMCONF_DESKTOP_MODE_REMOTE_DEVICES macro 1445

PRL_ERR_VMCONF_DUPLICATE_IP_ADDRESS 1445

PRL_ERR_VMCONF_DUPLICATE_IP_ADDRESS macro 1445

PRL_ERR_VMCONF_EFI_UNSUPPORTED_GUEST 1445

PRL_ERR_VMCONF_EFI_UNSUPPORTED_GUEST macro 1445

PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_EXIST 1445

PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_EXIST macro 1445

PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_VALID 1446

PRL_ERR_VMCONF_FLOPPY_DISK_IMAGE_IS_NOT_VALID macro 1446

PRL_ERR_VMCONF_FLOPPY_DISK_IS_NOT_ACCESSIBLE 1446

PRL_ERR_VMCONF_FLOPPY_DISK_IS_NOT_ACCESSIBLE macro 1446

PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_HAS_INVALID_SYMBOL
1446

PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_HAS_INVALID_SYMBOL
macro 1446

PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_IS_EMPTY 1446

PRL_ERR_VMCONF_FLOPPY_DISK_SYS_NAME_IS_EMPTY macro 1446

PRL_ERR_VMCONF_FLOPPY_DISK_URL_FORMAT_SYS_NAME
1447

PRL_ERR_VMCONF_FLOPPY_DISK_URL_FORMAT_SYS_NAME
macro 1447

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_CANNOT_BE_ADDED
1447

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_CANNOT_BE_ADDED
macro 1447

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_DUPLICATE_IN_ANOTHER_VM

1447

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_DUPLICATE_IN_ANOTHER_VM
macro 1447

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_CONNECTED
1448

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_CONNECTED
macro 1448

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_FOUND
1448

PRL_ERR_VMCONF_GENERIC_PCI_DEVICE_NOT_FOUND
macro 1448

PRL_ERR_VMCONF_GENERIC_PCI_DUPLICATE_SYS_NAME
1448

PRL_ERR_VMCONF_GENERIC_PCI_DUPLICATE_SYS_NAME
macro 1448

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE
1448

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_DEVICE_IS_ONE
macro 1448

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_NOT_SINGLE
1449

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_NOT_SINGLE
macro 1449

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_WRONG_COUNT
1449

PRL_ERR_VMCONF_GENERIC_PCI_VIDEO_WRONG_COUNT
macro 1449

PRL_ERR_VMCONF_GENERIC_PCI_WRONG_DEVICE 1449

PRL_ERR_VMCONF_GENERIC_PCI_WRONG_DEVICE
macro 1449

PRL_ERR_VMCONF_HARD_DISK_DUPLICATE_SYS_NAME
1450

PRL_ERR_VMCONF_HARD_DISK_DUPLICATE_SYS_NAME
macro 1450

PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_EXIST
1450

PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_EXIST
macro 1450

PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_VALID
1450

PRL_ERR_VMCONF_HARD_DISK_IMAGE_IS_NOT_VALID
macro 1450

PRL_ERR_VMCONF_HARD_DISK_MISS_BOOTCAMP_PARTITION
1450

PRL_ERR_VMCONF_HARD_DISK_MISS_BOOTCAMP_PARTITION
macro 1450

PRL_ERR_VMCONF_HARD_DISK_NOT_ENOUGH_SPACE_FOR_ENCRYPT_DISK
1451

PRL_ERR_VMCONF_HARD_DISK_NOT_ENOUGH_SPACE_FOR_ENCRYPT_DISK
macro 1451

PRL_ERR_VMCONF_HARD_DISK_SERIAL_IS_NOT_VALID
1451

PRL_ERR_VMCONF_HARD_DISK_SERIAL_IS_NOT_VALID
macro 1451

PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_OLD_CHIPSET
1451

PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_OLD_CHIPSET
macro 1451

PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_UNSUPPORTED_OS
1451

PRL_ERR_VMCONF_HARD_DISK_SET_SATA_FOR_UNSUPPORTED_OS
macro 1451

PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_HAS_INVALID_SYMBOL
1452

PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_HAS_INVALID_SYMBOL
macro 1452

PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_IS_EMPTY
1452

PRL_ERR_VMCONF_HARD_DISK_SYS_NAME_IS_EMPTY
macro 1452

PRL_ERR_VMCONF_HARD_DISK_UNABLE_DELETE_DISK_WITH_SNAPSHOTS
1452

PRL_ERR_VMCONF_HARD_DISK_UNABLE_DELETE_DISK_WITH_SNAPSHOTS
macro 1452

PRL_ERR_VMCONF_HARD_DISK_URL_FORMAT_SYS_NAME
1453

PRL_ERR_VMCONF_HARD_DISK_URL_FORMAT_SYS_NAME

macro 1453

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_BOOTCAMP_PARTITION
1453

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_BOOTCAMP_PARTITION
macro 1453

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_FOR_ENCRYPTRED_VM
1453

PRL_ERR_VMCONF_HARD_DISK_WRONG_TYPE_FOR_ENCRYPTRED_VM
macro 1453

PRL_ERR_VMCONF_IDE_DEVICES_COUNT_OUT_OF_RANGE
1453

PRL_ERR_VMCONF_IDE_DEVICES_COUNT_OUT_OF_RANGE
macro 1453

PRL_ERR_VMCONF_IDE_DEVICES_DUPLICATE_STACK_INDEX
1454

PRL_ERR_VMCONF_IDE_DEVICES_DUPLICATE_STACK_INDEX
macro 1454

PRL_ERR_VMCONF_INCOMPAT_HARD_DISK_SMART_GUARD_NOT_ALLOW
1454

PRL_ERR_VMCONF_INCOMPAT_HARD_DISK_SMART_GUARD_NOT_ALLOW
macro 1454

PRL_ERR_VMCONF_INCOMPAT_HARD_UNDO_DISKS_NOT_ALLOW
1454

PRL_ERR_VMCONF_INCOMPAT_HARD_UNDO_DISKS_NOT_ALLOW
macro 1454

PRL_ERR_VMCONF_INCOMPAT_SAFE_MODE_NOT_ALLOW
1455

PRL_ERR_VMCONF_INCOMPAT_SAFE_MODE_NOT_ALLOW
macro 1455

PRL_ERR_VMCONF_INVALID_DEVICE_MAIN_INDEX 1455

PRL_ERR_VMCONF_INVALID_DEVICE_MAIN_INDEX macro 1455

PRL_ERR_VMCONF_IOLIMIT_NOT_SUPPORTED 1455

PRL_ERR_VMCONF_IOLIMIT_NOT_SUPPORTED macro 1455

PRL_ERR_VMCONF_IOPRIO_NOT_SUPPORTED 1455

PRL_ERR_VMCONF_IOPRIO_NOT_SUPPORTED macro 1455

PRL_ERR_VMCONF_IOPSLIMIT_NOT_SUPPORTED 1456

PRL_ERR_VMCONF_IOPSLIMIT_NOT_SUPPORTED macro 1456

PRL_ERR_VMCONF_MAIN_MEMORY_MAX_BALLOON_SIZE_MORE_100_PERCENT
1456

PRL_ERR_VMCONF_MAIN_MEMORY_MAX_BALLOON_SIZE_MORE_100_PERCENT
macro 1456

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_INVALID_RANGE
1456

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_INVALID_RANGE
macro 1456

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_LESS_VMM_OVERHEAD_VALUE
1456

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_LESS_VMM_OVERHEAD_VALUE
macro 1456

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_OUT_OF_RANGE
1457

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_MIN_OUT_OF_RANGE
macro 1457

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_OUT_OF_RANGE
1457

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_OUT_OF_RANGE
macro 1457

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_ZERO
1457

PRL_ERR_VMCONF_MAIN_MEMORY_MQ_PRIOR_ZERO
macro 1457

PRL_ERR_VMCONF_MAIN_MEMORY_NOT_4_RATIO_SIZE
1457

PRL_ERR_VMCONF_MAIN_MEMORY_NOT_4_RATIO_SIZE
macro 1457

PRL_ERR_VMCONF_MAIN_MEMORY_OUT_OF_RANGE
1458

PRL_ERR_VMCONF_MAIN_MEMORY_OUT_OF_RANGE
macro 1458

PRL_ERR_VMCONF_MAIN_MEMORY_SIZE_NOT_EAQUAL_RECOMMENDED
1458

PRL_ERR_VMCONF_MAIN_MEMORY_SIZE_NOT_EAQUAL_RECOMMENDED
macro 1458

PRL_ERR_VMCONF_MAIN_MEMORY_ZERO_SIZE 1458

PRL_ERR_VMCONF_MAIN_MEMORY_ZERO_SIZE macro 1458

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTART 1458

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTART macro 1458

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTOP 1459

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_AUTOSTOP macro 1459

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_KEEP_VM_ALIVE_ON_GUI_EXIT 1459

PRL_ERR_VMCONF_NEED_HEADLESS_MODE_FOR_KEEP_VM_ALIVE_ON_GUI_EXIT macro 1459

PRL_ERR_VMCONF_NEED_MORE_MEMORY_TO_ENABLE_HOTPLUG 1459

PRL_ERR_VMCONF_NEED_MORE_MEMORY_TO_ENABLE_HOTPLUG macro 1459

PRL_ERR_VMCONF_NETWORK_ADAPTER_BROADCAST_IP_ADDRESS 1460

PRL_ERR_VMCONF_NETWORK_ADAPTER_BROADCAST_IP_ADDRESS macro 1460

PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_IP_ADDRESS 1460

PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_IP_ADDRESS macro 1460

PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_MAC_ADDRESS 1460

PRL_ERR_VMCONF_NETWORK_ADAPTER_DUPLICATE_MAC_ADDRESS macro 1460

PRL_ERR_VMCONF_NETWORK_ADAPTER_ETHLIST_CREATE_ERROR 1461

PRL_ERR_VMCONF_NETWORK_ADAPTER_ETHLIST_CREATE_ERROR macro 1461

PRL_ERR_VMCONF_NETWORK_ADAPTER_GATEWAY_NOT_IN_SUBNET 1461

PRL_ERR_VMCONF_NETWORK_ADAPTER_GATEWAY_NOT_IN_SUBNET macro 1461

PRL_ERR_VMCONF_NETWORK_ADAPTER_GUEST_TOOLS_NOT_AVAILABLE 1461

PRL_ERR_VMCONF_NETWORK_ADAPTER_GUEST_TOOLS_NOT_AVAILABLE macro 1461

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_BOUND_INDEX 1462

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_BOUND_INDEX macro 1462

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_DNS_IP_ADDRESS 1462

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_DNS_IP_ADDRESS macro 1462

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_GATEWAY_IP_ADDRESS 1462

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_GATEWAY_IP_ADDRESS macro 1462

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_IP_ADDRESS 1463

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_IP_ADDRESS macro 1463

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_MAC_ADDRESS 1463

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_MAC_ADDRESS macro 1463

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_SEARCH_DOMAIN_NAME 1463

PRL_ERR_VMCONF_NETWORK_ADAPTER_INVALID_SEARCH_DOMAIN_NAME macro 1463

PRL_ERR_VMCONF_NETWORK_ADAPTER_MULTICAST_IP_ADDRESS 1464

PRL_ERR_VMCONF_NETWORK_ADAPTER_MULTICAST_IP_ADDRESS
macro 1464
PRL_ERR_VMCONF_NETWORK_ADAPTER_ROUTED_NO_STATIC_ADDRESS
1464
PRL_ERR_VMCONF_NETWORK_ADAPTER_ROUTED_NO_STATIC_ADDRESS
macro 1464
PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_SMART_GUARD
1464
PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_SMART_GUARD
macro 1464
PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_UNDO_DISKS
1465
PRL_ERR_VMCONF_NO_AUTO_COMPRESS_WITH_UNDO_DISKS
macro 1465
PRL_ERR_VMCONF_NO_CONFIGURED_TOTALRATE_FOR_NETWORK_CLASS
1465
PRL_ERR_VMCONF_NO_CONFIGURED_TOTALRATE_FOR_NETWORK_CLASS
macro 1465
PRL_ERR_VMCONF_NO_HD_IMAGES_IN_SAFE_MODE
1465
PRL_ERR_VMCONF_NO_HD_IMAGES_IN_SAFE_MODE
macro 1465
PRL_ERR_VMCONF_NO_HD_IMAGES_IN_UNDO_DISKS_MODE
1465
PRL_ERR_VMCONF_NO_HD_IMAGES_IN_UNDO_DISKS_MODE
macro 1465
PRL_ERR_VMCONF_NO_IP_ADDRESSES_SPECIFIED_FOR_OFFLINE_MANAGEMENT
1466
PRL_ERR_VMCONF_NO_IP_ADDRESSES_SPECIFIED_FOR_OFFLINE_MANAGEMENT
macro 1466
PRL_ERR_VMCONF_NO_SMART_GUARD_WITH_UNDO_DISKS
1466
PRL_ERR_VMCONF_NO_SMART_GUARD_WITH_UNDO_DISKS
macro 1466
PRL_ERR_VMCONF_PARALLEL_PORT_IMAGE_IS_NOT_EXIST
1466
PRL_ERR_VMCONF_PARALLEL_PORT_IMAGE_IS_NOT_EXIST
macro 1466
PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
1467
PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
macro 1467
PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_IS_EMPTY
1467
PRL_ERR_VMCONF_PARALLEL_PORT_SYS_NAME_IS_EMPTY
macro 1467
PRL_ERR_VMCONF_PARALLEL_PORT_URL_FORMAT_SYS_NAME
1467
PRL_ERR_VMCONF_PARALLEL_PORT_URL_FORMAT_SYS_NAME
macro 1467
PRL_ERR_VMCONF_REAL_HARD_SAFE_MODE_NOT_ALLOW
1468
PRL_ERR_VMCONF_REAL_HARD_SAFE_MODE_NOT_ALLOW
macro 1468
PRL_ERR_VMCONF_REAL_HARD_UNDO_DISKS_NOT_ALLOW
1468
PRL_ERR_VMCONF_REAL_HARD_UNDO_DISKS_NOT_ALLOW
macro 1468
PRL_ERR_VMCONF_REMOTE_DISPLAY_EMPTY_PASSWORD
1468
PRL_ERR_VMCONF_REMOTE_DISPLAY_EMPTY_PASSWORD
macro 1468
PRL_ERR_VMCONF_REMOTE_DISPLAY_HOST_IP_ADDRESS_IS_ZERO
1469
PRL_ERR_VMCONF_REMOTE_DISPLAY_HOST_IP_ADDRESS_IS_ZERO
macro 1469
PRL_ERR_VMCONF_REMOTE_DISPLAY_INVALID_HOST_IP_ADDRESS
1469
PRL_ERR_VMCONF_REMOTE_DISPLAY_INVALID_HOST_IP_ADDRESS

macro 1469

PRL_ERR_VMCONF_REMOTE_DISPLAY_PASSWORD_TOO_LONG
1469

PRL_ERR_VMCONF_REMOTE_DISPLAY_PASSWORD_TOO_LONG
macro 1469

PRL_ERR_VMCONF_REMOTE_DISPLAY_PORT_NUMBER_IS_ZERO
1469

PRL_ERR_VMCONF_REMOTE_DISPLAY_PORT_NUMBER_IS_ZERO
macro 1469

PRL_ERR_VMCONF_RESTRICTED_CPU_COUNT 1470

PRL_ERR_VMCONF_RESTRICTED_CPU_COUNT macro 1470

PRL_ERR_VMCONF_RESTRICTED_MEMORY_SIZE 1470

PRL_ERR_VMCONF_RESTRICTED_MEMORY_SIZE macro 1470

PRL_ERR_VMCONF_RESTRICTED_OS_VERSION 1470

PRL_ERR_VMCONF_RESTRICTED_OS_VERSION macro 1470

PRL_ERR_VMCONF_RESTRICTED_SMART_GUARD 1470

PRL_ERR_VMCONF_RESTRICTED_SMART_GUARD macro 1470

PRL_ERR_VMCONF_RESTRICTED_UNDO_DISKS 1471

PRL_ERR_VMCONF_RESTRICTED_UNDO_DISKS macro 1471

PRL_ERR_VMCONF_SATA_DEVICES_COUNT_OUT_OF_RANGE
1471

PRL_ERR_VMCONF_SATA_DEVICES_COUNT_OUT_OF_RANGE
macro 1471

PRL_ERR_VMCONF_SATA_DEVICES_DUPLICATE_STACK_INDEX
1471

PRL_ERR_VMCONF_SATA_DEVICES_DUPLICATE_STACK_INDEX
macro 1471

PRL_ERR_VMCONF_SCSI_BUSLOGIC_WITH_EFI_NOT_SUPPORTED
1471

PRL_ERR_VMCONF_SCSI_BUSLOGIC_WITH_EFI_NOT_SUPPORTED
macro 1471

PRL_ERR_VMCONF_SCSI_DEVICES_COUNT_OUT_OF_RANGE
1472

PRL_ERR_VMCONF_SCSI_DEVICES_COUNT_OUT_OF_RANGE
macro 1472

PRL_ERR_VMCONF_SCSI_DEVICES_DUPLICATE_STACK_INDEX
1472

PRL_ERR_VMCONF_SCSI_DEVICES_DUPLICATE_STACK_INDEX
macro 1472

PRL_ERR_VMCONF_SERIAL_PORT_IMAGE_IS_NOT_EXIST
1472

PRL_ERR_VMCONF_SERIAL_PORT_IMAGE_IS_NOT_EXIST
macro 1472

PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
1473

PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_HAS_INVALID_SYMBOL
macro 1473

PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_IS_EMPTY
1473

PRL_ERR_VMCONF_SERIAL_PORT_SYS_NAME_IS_EMPTY
macro 1473

PRL_ERR_VMCONF_SERIAL_PORT_URL_FORMAT_SYS_NAME
1473

PRL_ERR_VMCONF_SERIAL_PORT_URL_FORMAT_SYS_NAME
macro 1473

PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_NAME
1474

PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_NAME
macro 1474

PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_PATH
1474

PRL_ERR_VMCONF_SHARED_FOLDERS_DUPLICATE_FOLDER_PATH
macro 1474

PRL_ERR_VMCONF_SHARED_FOLDERS_EMPTY_FOLDER_NAME
1474

PRL_ERR_VMCONF_SHARED_FOLDERS_EMPTY_FOLDER_NAME
macro 1474

PRL_ERR_VMCONF_SHARED_FOLDERS_INVALID_FOLDER_PATH

1475

PRL_ERR_VMCONF_SHARED_FOLDERS_INVALID_FOLDER_PATH
macro 1475

PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY 1475

PRL_ERR_VMCONF_SOUND_MIXER_IS_EMPTY macro 1475

PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY 1475

PRL_ERR_VMCONF_SOUND_OUTPUT_IS_EMPTY macro 1475

PRL_ERR_VMCONF_UNKNOWN_OS_TYPE 1475

PRL_ERR_VMCONF_UNKNOWN_OS_TYPE macro 1475

PRL_ERR_VMCONF_UNKNOWN_OS_VERSION 1476

PRL_ERR_VMCONF_UNKNOWN_OS_VERSION macro 1476

PRL_ERR_VMCONF_VALIDATION_FAILED 1476

PRL_ERR_VMCONF_VALIDATION_FAILED macro 1476

PRL_ERR_VMCONF_VIDEO_MEMORY_OUT_OF_RANGE 1476

PRL_ERR_VMCONF_VIDEO_MEMORY_OUT_OF_RANGE macro 1476

PRL_ERR_VMCONF_VIDEO_NOT_ENABLED 1476

PRL_ERR_VMCONF_VIDEO_NOT_ENABLED macro 1476

PRL_ERR_VMCONF_VIRTIO_BLOCK_DEVICES_DUPLICATE_STACK_INDEX
1477

PRL_ERR_VMCONF_VIRTIO_BLOCK_DEVICES_DUPLICATE_STACK_INDEX
macro 1477

PRL_ERR_VMCONF_VM_NAME_HAS_INVALID_SYMBOL 1477

PRL_ERR_VMCONF_VM_NAME_HAS_INVALID_SYMBOL macro 1477

PRL_ERR_VMCONF_VM_NAME_IS_EMPTY 1477

PRL_ERR_VMCONF_VM_NAME_IS_EMPTY macro 1477

PRL_ERR_VMCONF_VM_NAME_IS_TOO_LONG 1477

PRL_ERR_VMCONF_VM_NAME_IS_TOO_LONG macro 1477

PRL_ERR_VMDIR_INVALID_PATH 1478

PRL_ERR_VMDIR_INVALID_PATH macro 1478

PRL_ERR_VMDIR_PATH_IS_NOT_ABSOLUTE 1478

PRL_ERR_VMDIR_PATH_IS_NOT_ABSOLUTE macro 1478

PRL_ERR_VMHELPER_CREATEVM_FAILED 1478

PRL_ERR_VMHELPER_CREATEVM_FAILED macro 1478

PRL_ERR_VMHELPER_DIR_PATH_INVALID 1478

PRL_ERR_VMHELPER_DIR_PATH_INVALID macro 1478

PRL_ERR_VMHELPER_DISK_MOUNT_FAILED 1478

PRL_ERR_VMHELPER_DISK_MOUNT_FAILED macro 1478

PRL_ERR_VMHELPER_DISK_UNMOUNT_FAILED 1479

PRL_ERR_VMHELPER_DISK_UNMOUNT_FAILED macro 1479

PRL_ERR_VMHELPER_GETTING_AVAILABLE_DRIVE_FAILED
1479

PRL_ERR_VMHELPER_GETTING_AVAILABLE_DRIVE_FAILED
macro 1479

PRL_ERR_VNC_SERVER_ALREADY_STARTED 1479

PRL_ERR_VNC_SERVER_ALREADY_STARTED macro 1479

PRL_ERR_VNC_SERVER_AUTOSET_PORT_FAILED 1479

PRL_ERR_VNC_SERVER_AUTOSET_PORT_FAILED macro 1479

PRL_ERR_VNC_SERVER_DISABLED 1480

PRL_ERR_VNC_SERVER_DISABLED macro 1480

PRL_ERR_VNC_SERVER_NOT_STARTED 1480

PRL_ERR_VNC_SERVER_NOT_STARTED macro 1480

PRL_ERR_VOLUME_LICENSE_EXCEEDED_LIMIT 1480

PRL_ERR_VOLUME_LICENSE_EXCEEDED_LIMIT macro 1480

PRL_ERR_VTD_ALREADY_HOOKED_FAILED 1480

PRL_ERR_VTD_ALREADY_HOOKED_FAILED macro 1480

PRL_ERR_VTD_DEVICE_DOES_NOT_EXIST 1480

PRL_ERR_VTD_DEVICE_DOES_NOT_EXIST macro 1480

PRL_ERR_VTD_DEVICE_NOT_MAPPED 1481

PRL_ERR_VTD_DEVICE_NOT_MAPPED macro 1481

PRL_ERR_VTD_DEVICE_TROUBLESHOOT 1481

PRL_ERR_VTD_DEVICE_TROUBLESHOOT macro 1481

PRL_ERR_VTD_HOOK_AFTER_INSTALL_NEED_REBOOT 1481

PRL_ERR_VTD_HOOK_AFTER_INSTALL_NEED_REBOOT macro 1481

PRL_ERR_VTD_HOOK_AFTER_REVERT_NEED_REBOOT 1481

PRL_ERR_VTD_HOOK_AFTER_REVERT_NEED_REBOOT macro 1481

PRL_ERR_VTD_HOOK_DEVICE_CURRENTLY_IN_USE 1482

PRL_ERR_VTD_HOOK_DEVICE_CURRENTLY_IN_USE macro 1482

PRL_ERR_VTD_HOOK_FAILED 1482

PRL_ERR_VTD_HOOK_FAILED macro 1482

PRL_ERR_VTD_HOOK_INSTALLATION_FAILED 1482
PRL_ERR_VTD_HOOK_INSTALLATION_FAILED macro 1482
PRL_ERR_VTD_HOOK_INVALID_CONFIG 1482
PRL_ERR_VTD_HOOK_INVALID_CONFIG macro 1482
PRL_ERR_VTD_HOOK_NEED_REBOOT_FAILED 1482
PRL_ERR_VTD_HOOK_NEED_REBOOT_FAILED macro 1482
PRL_ERR_VTD_HOOK_REVERT_FAILED 1483
PRL_ERR_VTD_HOOK_REVERT_FAILED macro 1483
PRL_ERR_VTD_HOOK_UPDATE_SCRIPT_EXECUTE 1483
PRL_ERR_VTD_HOOK_UPDATE_SCRIPT_EXECUTE macro 1483
PRL_ERR_VTD_INITIALIZATION_FAILED 1483
PRL_ERR_VTD_INITIALIZATION_FAILED macro 1483
PRL_ERR_VTD_WAIT_ASR 1483
PRL_ERR_VTD_WAIT_ASR macro 1483
PRL_ERR_VTX_ENABLED_ONLY_IN_SMX 1483
PRL_ERR_VTX_ENABLED_ONLY_IN_SMX macro 1483
PRL_ERR_VZ_API_NOT_INITIALIZED 1484
PRL_ERR_VZ_API_NOT_INITIALIZED macro 1484
PRL_ERR_VZ_OPERATION_FAILED 1484
PRL_ERR_VZ_OPERATION_FAILED macro 1484
PRL_ERR_VZ_OSTEMPLATE_NOT_FOUND 1484
PRL_ERR_VZ_OSTEMPLATE_NOT_FOUND macro 1484
PRL_ERR_VZCTL_OPERATION_FAILED 1484
PRL_ERR_VZCTL_OPERATION_FAILED macro 1484
PRL_ERR_VZLICENSE_ACTIVATION_KEY 1484
PRL_ERR_VZLICENSE_ACTIVATION_KEY macro 1484
PRL_ERR_VZLICENSE_CANCEL 1485
PRL_ERR_VZLICENSE_CANCEL macro 1485
PRL_ERR_VZLICENSE_EACCES 1485
PRL_ERR_VZLICENSE_EACCES macro 1485
PRL_ERR_VZLICENSE_ERR_KA 1485
PRL_ERR_VZLICENSE_ERR_KA macro 1485
PRL_ERR_VZLICENSE_EXIST 1485
PRL_ERR_VZLICENSE_EXIST macro 1485
PRL_ERR_VZLICENSE_FATAL 1485
PRL_ERR_VZLICENSE_FATAL macro 1485
PRL_ERR_VZLICENSE_INVAL 1486
PRL_ERR_VZLICENSE_INVAL macro 1486
PRL_ERR_VZLICENSE_INVALID_HWID 1486
PRL_ERR_VZLICENSE_INVALID_HWID macro 1486
PRL_ERR_VZLICENSE_IO 1486
PRL_ERR_VZLICENSE_IO macro 1486
PRL_ERR_VZLICENSE_KA_ACTIVATION_LIMIT_REACHED 1486
PRL_ERR_VZLICENSE_KA_ACTIVATION_LIMIT_REACHED macro 1486
PRL_ERR_VZLICENSE_KA_HWID_DOES_NOT_MATCH 1487
PRL_ERR_VZLICENSE_KA_HWID_DOES_NOT_MATCH macro 1487
PRL_ERR_VZLICENSE_KA_LICENSE_EXPIRED 1487
PRL_ERR_VZLICENSE_KA_LICENSE_EXPIRED macro 1487
PRL_ERR_VZLICENSE_KA_LICENSE_IS_NOT_PROLONGATED_YET
1487
PRL_ERR_VZLICENSE_KA_LICENSE_IS_NOT_PROLONGATED_YET
macro 1487
PRL_ERR_VZLICENSE_KA_LICENSE_IS_TERMINATED 1487
PRL_ERR_VZLICENSE_KA_LICENSE_IS_TERMINATED macro 1487
PRL_ERR_VZLICENSE_KA_LICENSE_IS_UP_TO_DATE 1488
PRL_ERR_VZLICENSE_KA_LICENSE_IS_UP_TO_DATE macro 1488
PRL_ERR_VZLICENSE_LOCK 1488
PRL_ERR_VZLICENSE_LOCK macro 1488
PRL_ERR_VZLICENSE_NETWORK 1488
PRL_ERR_VZLICENSE_NETWORK macro 1488
PRL_ERR_VZLICENSE_NODATA 1488
PRL_ERR_VZLICENSE_NODATA macro 1488
PRL_ERR_VZLICENSE_NOENT 1488
PRL_ERR_VZLICENSE_NOENT macro 1488
PRL_ERR_VZLICENSE_NOINIT 1489
PRL_ERR_VZLICENSE_NOINIT macro 1489
PRL_ERR_VZLICENSE_NOMEM 1489
PRL_ERR_VZLICENSE_NOMEM macro 1489
PRL_ERR_VZLICENSE_NOTSUP 1489
PRL_ERR_VZLICENSE_NOTSUP macro 1489
PRL_ERR_VZLICENSE_PROXY 1489
PRL_ERR_VZLICENSE_PROXY macro 1489
PRL_ERR_VZLICENSE_PROXY_AUTH 1490
PRL_ERR_VZLICENSE_PROXY_AUTH macro 1490

PRL_ERR_VZLICENSE_TASK_ALREADY_RUN 1490
PRL_ERR_VZLICENSE_TASK_ALREADY_RUN macro 1490
PRL_ERR_VZLICENSE_TIMEOUT 1490
PRL_ERR_VZLICENSE_TIMEOUT macro 1490
PRL_ERR_VZLICENSE_UNSUPPORTED_APP_MODE 1490
PRL_ERR_VZLICENSE_UNSUPPORTED_APP_MODE macro 1490
PRL_ERR_VZLICENSE_UNSUPPORTED_MODE 1490
PRL_ERR_VZLICENSE_UNSUPPORTED_MODE macro 1490
PRL_ERR_VZLICENSE_UNSUPPORTED_UPDATE_OP 1491
PRL_ERR_VZLICENSE_UNSUPPORTED_UPDATE_OP macro 1491
PRL_ERR_VZLICENSE_VMS_LIMIT_EXCEEDED 1491
PRL_ERR_VZLICENSE_VMS_LIMIT_EXCEEDED macro 1491
PRL_ERR_VZLICENSE_WRONG 1491
PRL_ERR_VZLICENSE_WRONG macro 1491
PRL_ERR_WEB_PORTAL_ACCEPTED 1491
PRL_ERR_WEB_PORTAL_ACCEPTED macro 1491
PRL_ERR_WEB_PORTAL_ACCOUNT_ALREADY_EXISTS 1491
PRL_ERR_WEB_PORTAL_ACCOUNT_ALREADY_EXISTS macro 1491
PRL_ERR_WEB_PORTAL_BAD_REQUEST 1492
PRL_ERR_WEB_PORTAL_BAD_REQUEST macro 1492
PRL_ERR_WEB_PORTAL_CONFLICT 1492
PRL_ERR_WEB_PORTAL_CONFLICT macro 1492
PRL_ERR_WEB_PORTAL_CREATED 1492
PRL_ERR_WEB_PORTAL_CREATED macro 1492
PRL_ERR_WEB_PORTAL_FORBIDDEN 1492
PRL_ERR_WEB_PORTAL_FORBIDDEN macro 1492
PRL_ERR_WEB_PORTAL_INVALID_EMAIL 1493
PRL_ERR_WEB_PORTAL_INVALID_EMAIL macro 1493
PRL_ERR_WEB_PORTAL_INVALID_PARAMETERS 1493
PRL_ERR_WEB_PORTAL_INVALID_PARAMETERS macro 1493
PRL_ERR_WEB_PORTAL_INVALID_PASSWORD 1493
PRL_ERR_WEB_PORTAL_INVALID_PASSWORD macro 1493
PRL_ERR_WEB_PORTAL_NO_CONTENT 1493
PRL_ERR_WEB_PORTAL_NO_CONTENT macro 1493
PRL_ERR_WEB_PORTAL_NOT_FOUND 1493
PRL_ERR_WEB_PORTAL_NOT_FOUND macro 1493
PRL_ERR_WEB_PORTAL_OK 1494
PRL_ERR_WEB_PORTAL_OK macro 1494
PRL_ERR_WEB_PORTAL_SERVICE_UNAVAILABLE 1494
PRL_ERR_WEB_PORTAL_SERVICE_UNAVAILABLE macro 1494
PRL_ERR_WEB_PORTAL_SOCIAL_APP_DENIED 1494
PRL_ERR_WEB_PORTAL_SOCIAL_APP_DENIED macro 1494
PRL_ERR_WEB_PORTAL_SOCIAL_CONFIRM_EMAIL_SENT 1494
PRL_ERR_WEB_PORTAL_SOCIAL_CONFIRM_EMAIL_SENT macro 1494
PRL_ERR_WEB_PORTAL_SOCIAL_PROHIBITED_FOR_BA 1495
PRL_ERR_WEB_PORTAL_SOCIAL_PROHIBITED_FOR_BA macro 1495
PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_BAD_RESPONSE 1495
PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_BAD_RESPONSE macro 1495
PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_UNAVAILABLE 1495
PRL_ERR_WEB_PORTAL_SOCIAL_SERVICE_UNAVAILABLE macro 1495
PRL_ERR_WEB_PORTAL_SOCIAL_UNCONFIRMED_ACCOUNT 1495
PRL_ERR_WEB_PORTAL_SOCIAL_UNCONFIRMED_ACCOUNT macro 1495
PRL_ERR_WEB_PORTAL_UNAUTHORIZED 1496
PRL_ERR_WEB_PORTAL_UNAUTHORIZED macro 1496
PRL_ERR_WIN_AERO_DISABLED 1496
PRL_ERR_WIN_AERO_DISABLED macro 1496
PRL_ERR_WINDOWS_EXPRESS_INSTALL_USER_NAME_EMPTY 1496
PRL_ERR_WINDOWS_EXPRESS_INSTALL_USER_NAME_EMPTY macro 1496
PRL_ERR_WRITE_FAILED 1496
PRL_ERR_WRITE_FAILED macro 1496
PRL_ERR_WRONG_CONNECTION_SECURITY_LEVEL 1497

PRL_ERR_WRONG_CONNECTION_SECURITY_LEVEL macro 1497
PRL_ERR_WRONG_PROTOCOL_VERSION 1497
PRL_ERR_WRONG_PROTOCOL_VERSION macro 1497
PRL_ERR_WRONG_VM_STATE 1497
PRL_ERR_WRONG_VM_STATE macro 1497
PRL_ERR_WS_DISP_CONNECTION_CLOSED 1497
PRL_ERR_WS_DISP_CONNECTION_CLOSED macro 1497
PRL_ERR_X64GUEST_ON_X32HOST 1497
PRL_ERR_X64GUEST_ON_X32HOST macro 1497
PRL_ERR_X64GUEST_ON_X64VM_HVT_DISABLED 1498
PRL_ERR_X64GUEST_ON_X64VM_HVT_DISABLED macro 1498
PRL_ERR_XML_WRITE_FILE 1498
PRL_ERR_XML_WRITE_FILE macro 1498
PRL_ERR_XMLRPC_INTERNAL_ERROR 1498
PRL_ERR_XMLRPC_INTERNAL_ERROR macro 1498
PRL_ERR_XMLRPC_INVALID_CREDENTIALS 1498
PRL_ERR_XMLRPC_INVALID_CREDENTIALS macro 1498
PRL_ERR_XMLRPC_INVALID_REQUEST_INFO 1498
PRL_ERR_XMLRPC_INVALID_REQUEST_INFO macro 1498
PRL_ERR_XMLRPC_LIMITS_EXHAUSTED 1499
PRL_ERR_XMLRPC_LIMITS_EXHAUSTED macro 1499
PRL_ERR_XMLRPC_WRONG_METHOD_CALL 1499
PRL_ERR_XMLRPC_WRONG_METHOD_CALL macro 1499
PRL_EVENT_HANDLER_PTR 780
PRL_EVENT_ISSUER_TYPE 1053
PRL_EVENT_ISSUER_TYPE enumeration 1053
PRL_EVENT_ISSUER_TYPE_PTR 817
PRL_EVENT_ISSUER_TYPE_PTR type 817
PRL_EVENT_TYPE 1054
PRL_EVENT_TYPE enumeration 1054
PRL_EVENT_TYPE_PTR 817
PRL_EVENT_TYPE_PTR type 817
PRL_EXTERN 1625
PRL_EXTERN macro 1625
PRL_EXTERNAL_VM_TYPE 1058
PRL_EXTERNAL_VM_TYPE enumeration 1058
PRL_EXTERNAL_VM_TYPE_PTR 817
PRL_EXTERNAL_VM_TYPE_PTR type 817
PRL_FAILED 1626
PRL_FAILED macro 1626
PRL_FALSE 1626
PRL_FALSE macro 1626
PRL_FEATURES_MATRIX 1058
PRL_FEATURES_MATRIX enumeration 1058
PRL_FEATURES_MATRIX_PTR 818
PRL_FEATURES_MATRIX_PTR type 818
PRL_FILE Types 790
PRL_FILE_SYSTEM_ELEMENT_TYPE 1059
PRL_FILE_SYSTEM_ELEMENT_TYPE enumeration 1059
PRL_FILE_SYSTEM_ELEMENT_TYPE_PTR 791
PRL_FILE_SYSTEM_ELEMENT_TYPE_PTR type 791
PRL_FILE_SYSTEM_FS_TYPE 1059
PRL_FILE_SYSTEM_FS_TYPE enumeration 1059
PRL_FILE_SYSTEM_FS_TYPE_PTR 791
PRL_FILE_SYSTEM_FS_TYPE_PTR type 791
PRL_FILE_SYSTEM_TYPE 1060
PRL_FILE_SYSTEM_TYPE enumeration 1060
PRL_FILE_SYSTEM_TYPE_PTR 791
PRL_FILE_SYSTEM_TYPE_PTR type 791
PRL_FIREWALL_DIRECTION 1060
PRL_FIREWALL_DIRECTION enumeration 1060
PRL_FIREWALL_DIRECTION_PTR 818
PRL_FIREWALL_DIRECTION_PTR type 818
PRL_FIREWALL_POLICY 1060
PRL_FIREWALL_POLICY enumeration 1060
PRL_FIREWALL_POLICY_PTR 818
PRL_FIREWALL_POLICY_PTR type 818
PRL_FS_ADFS enumeration member 992
PRL_FS_AFFS enumeration member 992
PRL_FS_AFS enumeration member 992
PRL_FS_AUTOFS enumeration member 992
PRL_FS_CODA enumeration member 992
PRL_FS_EFS enumeration member 992
PRL_FS_EXTFS enumeration member 992
PRL_FS_FAT enumeration member 992
PRL_FS_FAT32 enumeration member 992
PRL_FS_FUSE enumeration member 992

PRL_FS_GFS enumeration member 992
PRL_FS_HFS enumeration member 992
PRL_FS_HPFS enumeration member 992
PRL_FS_INVALID enumeration member 992
PRL_FS_ISOFS enumeration member 992
PRL_FS_JFFS2 enumeration member 992
PRL_FS_NFS enumeration member 992
PRL_FS_NTFS enumeration member 992
PRL_FS_QNX4 enumeration member 992
PRL_FS_REISERFS enumeration member 992
PRL_FS_SMBFS enumeration member 992
PRL_FS_UNSPECIFIED enumeration member 992
PRL_GENERIC_DEVICE_STATE 1061
PRL_GENERIC_DEVICE_STATE enumeration 1061
PRL_GENERIC_DEVICE_STATE_PTR 818
PRL_GENERIC_DEVICE_STATE_PTR type 818
PRL_GENERIC_PCI_DEVICE_CLASS 1061
PRL_GENERIC_PCI_DEVICE_CLASS enumeration 1061
PRL_GENERIC_PCI_DEVICE_CLASS_PTR 818
PRL_GENERIC_PCI_DEVICE_CLASS_PTR type 818
PRL_GET_SNAPSHOT_TREE_FLAGS 1061
PRL_GET_SNAPSHOT_TREE_FLAGS enumeration 1061
PRL_GET_VM_CONFIG_FLAGS 1062
PRL_GET_VM_CONFIG_FLAGS enumeration 1062
PRL_GET_VM_CONFIG_FLAGS_PTR 819
PRL_GET_VM_CONFIG_FLAGS_PTR type 819
PRL_GET_VM_LIST_FLAGS 1062
PRL_GET_VM_LIST_FLAGS enumeration 1062
PRL_GET_VM_LIST_FLAGS_PTR 819
PRL_GET_VM_LIST_FLAGS_PTR type 819
PRL_GRACEFUL_SHUTDOWN_CAPABILITIES 1062
PRL_GRACEFUL_SHUTDOWN_CAPABILITIES enumeration 1062
PRL_GRACEFUL_SHUTDOWN_CAPABILITIES_PTR 819
PRL_GRACEFUL_SHUTDOWN_CAPABILITIES_PTR type 819
PRL_GRACEFUL_SHUTDOWN_KIND 1063
PRL_GRACEFUL_SHUTDOWN_KIND enumeration 1063
PRL_GRACEFUL_SHUTDOWN_KIND_PTR 819
PRL_GRACEFUL_SHUTDOWN_KIND_PTR type 819
PRL_GUEST Macros 1499
PRL_GUEST_CPU_TIME_PTRN 1500
PRL_GUEST_CPU_TIME_PTRN macro 1500
PRL_GUEST_CPU_USAGE_PTRN 1500
PRL_GUEST_CPU_USAGE_PTRN macro 1500
PRL_GUEST_DISTRO_VERSION_ENUM 1063
PRL_GUEST_DISTRO_VERSION_ENUM enumeration 1063
PRL_GUEST_DISTRO_VERSION_ENUM_PTR 819
PRL_GUEST_DISTRO_VERSION_ENUM_PTR type 819
PRL_GUEST_FS_DISK_INDEX_PTRN 1500
PRL_GUEST_FS_DISK_INDEX_PTRN macro 1500
PRL_GUEST_FS_FREE_SPACE_PTRN 1500
PRL_GUEST_FS_FREE_SPACE_PTRN macro 1500
PRL_GUEST_FS_NAME_PTRN 1501
PRL_GUEST_FS_NAME_PTRN macro 1501
PRL_GUEST_FS_TOTAL_SPACE_PTRN 1501
PRL_GUEST_FS_TOTAL_SPACE_PTRN macro 1501
PRL_GUEST_OS_SUPPORT_TYPE 1063
PRL_GUEST_OS_SUPPORT_TYPE enumeration 1063
PRL_GUEST_OS_SUPPORT_TYPE_PTR 820
PRL_GUEST_OS_SUPPORT_TYPE_PTR type 820
PRL_GUEST_RAM_BALLOON_ACTUAL 1501
PRL_GUEST_RAM_BALLOON_ACTUAL macro 1501
PRL_GUEST_RAM_CACHED_PTRN 1501
PRL_GUEST_RAM_CACHED_PTRN macro 1501
PRL_GUEST_RAM_MAJOR_FAULT 1502
PRL_GUEST_RAM_MAJOR_FAULT macro 1502
PRL_GUEST_RAM_MINOR_FAULT 1502
PRL_GUEST_RAM_MINOR_FAULT macro 1502
PRL_GUEST_RAM_SWAP_IN 1502
PRL_GUEST_RAM_SWAP_IN macro 1502
PRL_GUEST_RAM_SWAP_OUT 1502
PRL_GUEST_RAM_SWAP_OUT macro 1502
PRL_GUEST_RAM_TOTAL_PTRN 1502
PRL_GUEST_RAM_TOTAL_PTRN macro 1502
PRL_GUEST_RAM_USAGE_PTRN 1503
PRL_GUEST_RAM_USAGE_PTRN macro 1503
PRL_GUEST_VCPU_TIME_PTRN 1503
PRL_GUEST_VCPU_TIME_PTRN macro 1503
PRL_HANDLE 820

PRL_HANDLE type 820
PRL_HANDLE_PTR 820
PRL_HANDLE_PTR type 820
PRL_HANDLE_TYPE 1064
PRL_HANDLE_TYPE enumeration 1064
PRL_HANDLE_TYPE_PTR 820
PRL_HANDLE_TYPE_PTR type 820
PRL_HARD_DISK_IFACE_MASK 1065
PRL_HARD_DISK_IFACE_MASK enumeration 1065
PRL_HARD_DISK_IFACE_MASK_PTR 820
PRL_HARD_DISK_IFACE_MASK_PTR type 820
PRL_HARD_DISK_INTERNAL_FORMAT 1066
PRL_HARD_DISK_INTERNAL_FORMAT enumeration 1066
PRL_HARD_DISK_INTERNAL_FORMAT_PTR 821
PRL_HARD_DISK_INTERNAL_FORMAT_PTR type 821
PRL_HOST_CPU_TIME_PTRN 1626
PRL_HOST_CPU_TIME_PTRN macro 1626
PRL_HOST_OS_TYPE 1066
PRL_HOST_OS_TYPE enumeration 1066
PRL_HOST_OS_TYPE_PTR 821
PRL_HOST_OS_TYPE_PTR type 821
PRL_HW_INFO_NET_ADAPTER_TYPE 1067
PRL_HW_INFO_NET_ADAPTER_TYPE enumeration 1067
PRL_HW_INFO_NET_ADAPTER_TYPE_PTR 821
PRL_HW_INFO_NET_ADAPTER_TYPE_PTR type 821
PRL_IDE_READ_REQUESTS_PTRN 1626
PRL_IDE_READ_REQUESTS_PTRN macro 1626
PRL_IDE_READ_TOTAL_PTRN 1626
PRL_IDE_READ_TOTAL_PTRN macro 1626
PRL_IDE_WRITE_REQUESTS_PTRN 1627
PRL_IDE_WRITE_REQUESTS_PTRN macro 1627
PRL_IDE_WRITE_TOTAL_PTRN 1627
PRL_IDE_WRITE_TOTAL_PTRN macro 1627
PRL_IMAGE Macros 1503
PRL_IMAGE_BOOTCAMP 1504
PRL_IMAGE_BOOTCAMP macro 1504
PRL_IMAGE_BOOTCAMP_FAKE 1504
PRL_IMAGE_BOOTCAMP_FAKE macro 1504
PRL_IMAGE_BOOTCAMP_UID 1504
PRL_IMAGE_BOOTCAMP_UID macro 1504
PRL_IMAGE_COMPRESSED 1504
PRL_IMAGE_COMPRESSED macro 1504
PRL_IMAGE_FORMAT 1067
PRL_IMAGE_FORMAT enumeration 1067
PRL_IMAGE_FORMAT_PTR 821
PRL_IMAGE_FORMAT_PTR type 821
PRL_IMAGE_INVALID 1505
PRL_IMAGE_INVALID macro 1505
PRL_IMAGE_LAST_INVALID 1505
PRL_IMAGE_LAST_INVALID macro 1505
PRL_IMAGE_PARAMETERS 1067
PRL_IMAGE_PARAMETERS structure 1067
PRL_IMAGE_PARAMETERS_PTR 822
PRL_IMAGE_PARAMETERS_PTR type 822
PRL_IMAGE_PARTITION 1505
PRL_IMAGE_PARTITION macro 1505
PRL_IMAGE_PHYSICAL 1505
PRL_IMAGE_PHYSICAL macro 1505
PRL_IMAGE_PLAIN 1506
PRL_IMAGE_PLAIN macro 1506
PRL_IMAGE_TRY_GUESS 1506
PRL_IMAGE_TRY_GUESS macro 1506
PRL_IMAGE_TYPE 822
PRL_IMAGE_TYPE type 822
PRL_IMAGE_TYPE_PTR 822
PRL_IMAGE_TYPE_PTR type 822
PRL_IMAGE_VHD_DIFF 1506
PRL_IMAGE_VHD_DIFF macro 1506
PRL_IMAGE_VHD_DYNAMIC 1506
PRL_IMAGE_VHD_DYNAMIC macro 1506
PRL_IMAGE_VHD_DYNAMIC_PART 1507
PRL_IMAGE_VHD_DYNAMIC_PART macro 1507
PRL_IMAGE_VHD_PLAIN 1507
PRL_IMAGE_VHD_PLAIN macro 1507
PRL_IMAGE_VHD_PLAIN_PART 1507
PRL_IMAGE_VHD_PLAIN_PART macro 1507
PRL_IMAGE_VMDK_EXTENT_2GB_FLAT 1507
PRL_IMAGE_VMDK_EXTENT_2GB_FLAT macro 1507

PRL_IMAGE_VMDK_EXTENT_2GB_SPARSE 1507
PRL_IMAGE_VMDK_EXTENT_2GB_SPARSE macro 1507
PRL_IMAGE_VMDK_MONOLITHIC_FLAT 1508
PRL_IMAGE_VMDK_MONOLITHIC_FLAT macro 1508
PRL_IMAGE_VMDK_MONOLITHIC_SPARSE 1508
PRL_IMAGE_VMDK_MONOLITHIC_SPARSE macro 1508
PRL_INFO_VM_MIGRATE_STORAGE_IS_SHARED 1627
PRL_INFO_VM_MIGRATE_STORAGE_IS_SHARED macro 1627
PRL_INSTALL_APPLIANCE_FLAGS 1068
PRL_INSTALL_APPLIANCE_FLAGS enumeration 1068
PRL_INT Types 791
PRL_INT16 792
PRL_INT16 type 792
PRL_INT16_PTR 792
PRL_INT16_PTR type 792
PRL_INT16_PTR_PTR 792
PRL_INT16_PTR_PTR type 792
PRL_INT32 792
PRL_INT32 type 792
PRL_INT32_PTR 793
PRL_INT32_PTR type 793
PRL_INT32_PTR_PTR 793
PRL_INT32_PTR_PTR type 793
PRL_INT64 793
PRL_INT64 type 793
PRL_INT64_PTR 793
PRL_INT64_PTR type 793
PRL_INT64_PTR_PTR 793
PRL_INT64_PTR_PTR type 793
PRL_INT8 794
PRL_INT8 type 794
PRL_INT8_PTR 794
PRL_INT8_PTR type 794
PRL_INT8_PTR_PTR 794
PRL_INT8_PTR_PTR type 794
PRL_INVALID_HANDLE 1627
PRL_INVALID_HANDLE macro 1627
PRL_INVALID_PARTITION 1628
PRL_INVALID_PARTITION macro 1628
PRL_INVALID_VLAN_TAG 1628
PRL_INVALID_VLAN_TAG macro 1628
PRL_IO Structs 837
PRL_IO Types 794
PRL_IO_CLIENT_FLAGS 918
PRL_IO_CLIENT_FLAGS enumeration 918
PRL_IO_DISCONNECT_REASON 918
PRL_IO_DISCONNECT_REASON enumeration 918
PRL_IO_DISCONNECT_REASON_PTR 795
PRL_IO_DISCONNECT_REASON_PTR type 795
PRL_IO_DISPLAY_SCREEN_SIZE 919
PRL_IO_DISPLAY_SCREEN_SIZE structure 919
PRL_IO_DISPLAY_SCREEN_SIZE_PTR 795
PRL_IO_DISPLAY_SCREEN_SIZE_PTR type 795
PRL_IO_GAMMARAMP_DEFAULT 1628
PRL_IO_GAMMARAMP_DEFAULT macro 1628
PRL_IO_GAMMARAMP_RGB256x3x16 1628
PRL_IO_GAMMARAMP_RGB256x3x16 macro 1628
PRL_IO_MAX_DISPLAYS 1628
PRL_IO_MAX_DISPLAYS macro 1628
PRL_IO_MAX_ENCODINGS 1629
PRL_IO_MAX_ENCODINGS macro 1629
PRL_IO_PALETTE 795
PRL_IO_PALETTE type 795
PRL_IO_SCREEN_SIZE 919
PRL_IO_SCREEN_SIZE structure 919
PRL_IO_SCREEN_SIZE_PTR 795
PRL_IO_SCREEN_SIZE_PTR type 795
PRL_IO_STATE 919
PRL_IO_STATE enumeration 919
PRL_IO_STATE_PTR 795
PRL_IO_STATE_PTR type 795
PRL_IOLIMIT_BS enumeration member 1000
PRL_IOLIMIT_TYPE 920
PRL_IOLIMIT_TYPE enumeration 920
PRL_IOPRIO 1068
PRL_IOPRIO enumeration 1068
PRL_IOPRIO_DEFAULT enumeration member 1001
PRL_IOPRIO_MAX enumeration member 1001

PRL_IOPRIO_MIN enumeration member 1001
PRL_ISCSI_STORAGE_CMD_FLAGS 1068
PRL_ISCSI_STORAGE_CMD_FLAGS enumeration 1068
PRL_ISCSI_STORAGE_FSTYPE_FLAGS 1069
PRL_ISCSI_STORAGE_FSTYPE_FLAGS enumeration 1069
PRL_JOB_OPERATION_CODE 1069
PRL_JOB_OPERATION_CODE enumeration 1069
PRL_JOB_OPERATION_CODE_PTR 822
PRL_JOB_OPERATION_CODE_PTR type 822
PRL_JOB_STATUS 1072
PRL_JOB_STATUS enumeration 1072
PRL_JOB_STATUS_PTR 822
PRL_JOB_STATUS_PTR type 822
PRL_KEY 1073
PRL_KEY enumeration 1073
PRL_KEY_0 enumeration member 1005
PRL_KEY_1 enumeration member 1005
PRL_KEY_2 enumeration member 1005
PRL_KEY_3 enumeration member 1005
PRL_KEY_4 enumeration member 1005
PRL_KEY_5 enumeration member 1005
PRL_KEY_6 enumeration member 1005
PRL_KEY_7 enumeration member 1005
PRL_KEY_8 enumeration member 1005
PRL_KEY_9 enumeration member 1005
PRL_KEY_A enumeration member 1005
PRL_KEY_APP_CALCULATOR enumeration member 1005
PRL_KEY_APP_DASHBOARD enumeration member 1005
PRL_KEY_APP_EXPOSE enumeration member 1005
PRL_KEY_APP_MAIL enumeration member 1005
PRL_KEY_APP_MY_COMPUTER enumeration member 1005
PRL_KEY_B enumeration member 1005
PRL_KEY_BACKSLASH enumeration member 1005
PRL_KEY_BACKSPACE enumeration member 1005
PRL_KEY_BRAZILIAN_KEYPAD enumeration member 1005
PRL_KEY_BREAK enumeration member 1005
PRL_KEY_C enumeration member 1005
PRL_KEY_CAPS_LOCK enumeration member 1005
PRL_KEY_COMMA enumeration member 1005
PRL_KEY_D enumeration member 1005
PRL_KEY_DELETE enumeration member 1005
PRL_KEY_DOLLAR enumeration member 1005
PRL_KEY_DOT enumeration member 1005
PRL_KEY_DOWN enumeration member 1005
PRL_KEY_E enumeration member 1005
PRL_KEY_EJECT enumeration member 1005
PRL_KEY_END enumeration member 1005
PRL_KEY_ENTER enumeration member 1005
PRL_KEY_EQUAL enumeration member 1005
PRL_KEY_ESCAPE enumeration member 1005
PRL_KEY_EURO enumeration member 1005
PRL_KEY_EUROPE_1 enumeration member 1005
PRL_KEY_EUROPE_2 enumeration member 1005
PRL_KEY_EVENT 1076
PRL_KEY_EVENT enumeration 1076
PRL_KEY_EVENT_PTR 823
PRL_KEY_EVENT_PTR type 823
PRL_KEY_F enumeration member 1005
PRL_KEY_F1 enumeration member 1005
PRL_KEY_F10 enumeration member 1005
PRL_KEY_F11 enumeration member 1005
PRL_KEY_F12 enumeration member 1005
PRL_KEY_F13 enumeration member 1005
PRL_KEY_F14 enumeration member 1005
PRL_KEY_F15 enumeration member 1005
PRL_KEY_F16 enumeration member 1005
PRL_KEY_F17 enumeration member 1005
PRL_KEY_F18 enumeration member 1005
PRL_KEY_F19 enumeration member 1005
PRL_KEY_F2 enumeration member 1005
PRL_KEY_F20 enumeration member 1005
PRL_KEY_F21 enumeration member 1005
PRL_KEY_F22 enumeration member 1005
PRL_KEY_F23 enumeration member 1005
PRL_KEY_F24 enumeration member 1005
PRL_KEY_F3 enumeration member 1005
PRL_KEY_F4 enumeration member 1005
PRL_KEY_F5 enumeration member 1005

PRL_KEY_F6 enumeration member 1005
PRL_KEY_F7 enumeration member 1005
PRL_KEY_F8 enumeration member 1005
PRL_KEY_F9 enumeration member 1005
PRL_KEY_FN enumeration member 1005
PRL_KEY_G enumeration member 1005
PRL_KEY_H enumeration member 1005
PRL_KEY_HANGUEL enumeration member 1005
PRL_KEY_HANJA enumeration member 1005
PRL_KEY_HENKAN enumeration member 1005
PRL_KEY_HIRAGANA enumeration member 1005
PRL_KEY_HIRAGANA_KATAKANA enumeration member 1005
PRL_KEY_HOME enumeration member 1005
PRL_KEY_I enumeration member 1005
PRL_KEY_INSERT enumeration member 1005
PRL_KEY_INVALID enumeration member 1005
PRL_KEY_J enumeration member 1005
PRL_KEY_K enumeration member 1005
PRL_KEY_KATAKANA enumeration member 1005
PRL_KEY_KBD_BRIGHTNESS_DOWN enumeration member 1005
PRL_KEY_KBD_BRIGHTNESS_UP enumeration member 1005
PRL_KEY_L enumeration member 1005
PRL_KEY_LEFT enumeration member 1005
PRL_KEY_LEFT_ALT enumeration member 1005
PRL_KEY_LEFT_BRACKET enumeration member 1005
PRL_KEY_LEFT_BUTTON enumeration member 1005
PRL_KEY_LEFT_CONTROL enumeration member 1005
PRL_KEY_LEFT_SHIFT enumeration member 1005
PRL_KEY_LEFT_WIN enumeration member 1005
PRL_KEY_M enumeration member 1005
PRL_KEY_MAC129 enumeration member 1005
PRL_KEY_MAX enumeration member 1005
PRL_KEY_MEDIA_NEXT_TRACK enumeration member 1005
PRL_KEY_MEDIA_PLAY_PAUSE enumeration member 1005
PRL_KEY_MEDIA_PREV_TRACK enumeration member 1005
PRL_KEY_MEDIA_SELECT enumeration member 1005
PRL_KEY_MEDIA_STOP enumeration member 1005
PRL_KEY_MENU enumeration member 1005
PRL_KEY_MIDDLE_BUTTON enumeration member 1005
PRL_KEY_MINUS enumeration member 1005
PRL_KEY_MON_BRIGHTNESS_DOWN enumeration member 1005
PRL_KEY_MON_BRIGHTNESS_UP enumeration member 1005
PRL_KEY_MOVE_DOWN enumeration member 1005
PRL_KEY_MOVE_DOWN_LEFT enumeration member 1005
PRL_KEY_MOVE_DOWN_RIGHT enumeration member 1005
PRL_KEY_MOVE_LEFT enumeration member 1005
PRL_KEY_MOVE_RIGHT enumeration member 1005
PRL_KEY_MOVE_UP enumeration member 1005
PRL_KEY_MOVE_UP_LEFT enumeration member 1005
PRL_KEY_MOVE_UP_RIGHT enumeration member 1005
PRL_KEY_MUHENKAN enumeration member 1005
PRL_KEY_MUTE enumeration member 1005
PRL_KEY_N enumeration member 1005
PRL_KEY_NP_0 enumeration member 1005
PRL_KEY_NP_1 enumeration member 1005
PRL_KEY_NP_2 enumeration member 1005
PRL_KEY_NP_3 enumeration member 1005
PRL_KEY_NP_4 enumeration member 1005
PRL_KEY_NP_5 enumeration member 1005
PRL_KEY_NP_6 enumeration member 1005
PRL_KEY_NP_7 enumeration member 1005
PRL_KEY_NP_8 enumeration member 1005
PRL_KEY_NP_9 enumeration member 1005
PRL_KEY_NP_DELETE enumeration member 1005
PRL_KEY_NP_ENTER enumeration member 1005
PRL_KEY_NP_EQUAL enumeration member 1005
PRL_KEY_NP_MINUS enumeration member 1005
PRL_KEY_NP_PLUS enumeration member 1005
PRL_KEY_NP_SLASH enumeration member 1005
PRL_KEY_NP_STAR enumeration member 1005
PRL_KEY_NUM_LOCK enumeration member 1005
PRL_KEY_O enumeration member 1005
PRL_KEY_P enumeration member 1005
PRL_KEY_PAGE_DOWN enumeration member 1005
PRL_KEY_PAGE_UP enumeration member 1005
PRL_KEY_PAUSE enumeration member 1005
PRL_KEY_PC9800_KEYPAD enumeration member 1005

PRL_KEY_PRINT enumeration member 1005
PRL_KEY_PRINT_WITH_MODIFIER enumeration member 1005
PRL_KEY_Q enumeration member 1005
PRL_KEY_QUOTE enumeration member 1005
PRL_KEY_R enumeration member 1005
PRL_KEY_RIGHT enumeration member 1005
PRL_KEY_RIGHT_ALT enumeration member 1005
PRL_KEY_RIGHT_BRACKET enumeration member 1005
PRL_KEY_RIGHT_BUTTON enumeration member 1005
PRL_KEY_RIGHT_CONTROL enumeration member 1005
PRL_KEY_RIGHT_SHIFT enumeration member 1005
PRL_KEY_RIGHT_WIN enumeration member 1005
PRL_KEY_RO enumeration member 1005
PRL_KEY_S enumeration member 1005
PRL_KEY_SCROLL_LOCK enumeration member 1005
PRL_KEY_SEMICOLON enumeration member 1005
PRL_KEY_SLASH enumeration member 1005
PRL_KEY_SPACE enumeration member 1005
PRL_KEY_SYSRQ enumeration member 1005
PRL_KEY_SYSTEM_POWER enumeration member 1005
PRL_KEY_SYSTEM_SLEEP enumeration member 1005
PRL_KEY_SYSTEM_WAKE enumeration member 1005
PRL_KEY_T enumeration member 1005
PRL_KEY_TAB enumeration member 1005
PRL_KEY_TILDA enumeration member 1005
PRL_KEY_U enumeration member 1005
PRL_KEY_UP enumeration member 1005
PRL_KEY_V enumeration member 1005
PRL_KEY_VOLUME_DOWN enumeration member 1005
PRL_KEY_VOLUME_UP enumeration member 1005
PRL_KEY_W enumeration member 1005
PRL_KEY_WHEEL_DOWN enumeration member 1005
PRL_KEY_WHEEL_LEFT enumeration member 1005
PRL_KEY_WHEEL_RIGHT enumeration member 1005
PRL_KEY_WHEEL_UP enumeration member 1005
PRL_KEY_WILDCARD_ALT enumeration member 1005
PRL_KEY_WILDCARD_ANY enumeration member 1005
PRL_KEY_WILDCARD_CTRL enumeration member 1005
PRL_KEY_WILDCARD_KEYBOARD enumeration member 1005
PRL_KEY_WILDCARD_MOUSE enumeration member 1005
PRL_KEY_WILDCARD_SHIFT enumeration member 1005
PRL_KEY_WILDCARD_WIN enumeration member 1005
PRL_KEY_WWW_BACK enumeration member 1005
PRL_KEY_WWW_FAVORITES enumeration member 1005
PRL_KEY_WWW_FORWARD enumeration member 1005
PRL_KEY_WWW_HOME enumeration member 1005
PRL_KEY_WWW_REFRESH enumeration member 1005
PRL_KEY_WWW_SEARCH enumeration member 1005
PRL_KEY_WWW_STOP enumeration member 1005
PRL_KEY_X enumeration member 1005
PRL_KEY_Y enumeration member 1005
PRL_KEY_YEN enumeration member 1005
PRL_KEY_Z enumeration member 1005
PRL_KEY_ZENKAKU_HANKAKU enumeration member 1005
PRL_LICENSE_IS_EXPIRED 1629
PRL_LICENSE_IS_EXPIRED macro 1629
PRL_LICENSE_IS_VALID 1629
PRL_LICENSE_IS_VALID macro 1629
PRL_LICENSE_RESTRICTION_KEY 1076
PRL_LICENSE_RESTRICTION_KEY enumeration 1076
PRL_LICENSE_RESTRICTION_KEY_PTR 823
PRL_LICENSE_RESTRICTION_KEY_PTR type 823
PRL_LINUX_EXTENDED_PARTITION 1629
PRL_LINUX_EXTENDED_PARTITION macro 1629
PRL_LINUX_PARTITION_DATA 1630
PRL_LINUX_PARTITION_DATA macro 1630
PRL_LINUX_PARTITION_SWAP 1630
PRL_LINUX_PARTITION_SWAP macro 1630
PRL_LOGIN_LOCAL_FLAGS 1078
PRL_LOGIN_LOCAL_FLAGS enumeration 1078
PRL_LOGIN_LOCAL_FLAGS_PTR 823
PRL_LOGIN_LOCAL_FLAGS_PTR type 823
PRL_MACOS_BOOT_PARTITION 1630
PRL_MACOS_BOOT_PARTITION macro 1630
PRL_MACOS_PARTITION 1630
PRL_MACOS_PARTITION macro 1630
PRL_MASS_STORAGE_INTERFACE_TYPE 1078

PRL_MASS_STORAGE_INTERFACE_TYPE enumeration 1078
PRL_MASS_STORAGE_INTERFACE_TYPE_PTR 823
PRL_MASS_STORAGE_INTERFACE_TYPE_PTR type 823
PRL_MAX_VLAN_TAG 1630
PRL_MAX_VLAN_TAG macro 1630
PRL_MEMGUARANTEE_AUTO enumeration member 1011
PRL_MEMGUARANTEE_PERCENTS enumeration member 1011
PRL_MEMGUARANTEE_TYPE 1079
PRL_MEMGUARANTEE_TYPE enumeration 1079
PRL_MESSAGE_TYPE_ENUM 1079
PRL_MESSAGE_TYPE_ENUM enumeration 1079
PRL_MESSAGE_TYPE_ENUM_PTR 824
PRL_MESSAGE_TYPE_ENUM_PTR type 824
PRL_METHOD 1631
PRL_METHOD macro 1631
PRL_METHOD_PTR 1631
PRL_METHOD_PTR macro 1631
PRL_MOBILE_ADVANCED_AUTH_MODE 1079
PRL_MOBILE_ADVANCED_AUTH_MODE enumeration 1079
PRL_MOBILE_ADVANCED_AUTH_MODE_PTR 824
PRL_MOBILE_ADVANCED_AUTH_MODE_PTR type 824
PRL_MOUNT_VIRTUAL_DISK 1079
PRL_MOUNT_VIRTUAL_DISK enumeration 1079
PRL_MOUSE_BUTTON 1080
PRL_MOUSE_BUTTON enumeration 1080
PRL_MOUSE_BUTTON_PTR 824
PRL_MOUSE_BUTTON_PTR type 824
PRL_MSDOS_EXTENDED_PARTITION 1631
PRL_MSDOS_EXTENDED_PARTITION macro 1631
PRL_NET Macros 1508
PRL_NET_ADAPTER_ALREADY_USED 1510
PRL_NET_ADAPTER_ALREADY_USED macro 1510
PRL_NET_ADAPTER_EMULATED_TYPE 1080
PRL_NET_ADAPTER_EMULATED_TYPE enumeration 1080
PRL_NET_ADAPTER_EMULATED_TYPE_PTR 824
PRL_NET_ADAPTER_EMULATED_TYPE_PTR type 824
PRL_NET_ADAPTER_NOT_EXIST 1510
PRL_NET_ADAPTER_NOT_EXIST macro 1510
PRL_NET_ADAPTER_TYPE 1081
PRL_NET_ADAPTER_TYPE enumeration 1081
PRL_NET_ADAPTER_TYPE_PTR 824
PRL_NET_ADAPTER_TYPE_PTR type 824
PRL_NET_BIND_FAILED 1510
PRL_NET_BIND_FAILED macro 1510
PRL_NET_BYTES_IN_PTRN 1510
PRL_NET_BYTES_IN_PTRN macro 1510
PRL_NET_BYTES_OUT_PTRN 1511
PRL_NET_BYTES_OUT_PTRN macro 1511
PRL_NET_CABLE_DISCONNECTED 1511
PRL_NET_CABLE_DISCONNECTED macro 1511
PRL_NET_CLASSFUL_TRAFFIC_IPV4_PTRN 1511
PRL_NET_CLASSFUL_TRAFFIC_IPV4_PTRN macro 1511
PRL_NET_CLASSFUL_TRAFFIC_IPV6_PTRN 1511
PRL_NET_CLASSFUL_TRAFFIC_IPV6_PTRN macro 1511
PRL_NET_CLASSFUL_TRAFFIC_PTRN 1511
PRL_NET_CLASSFUL_TRAFFIC_PTRN macro 1511
PRL_NET_CONNECTION_SHARING_CONFLICT 1512
PRL_NET_CONNECTION_SHARING_CONFLICT macro 1512
PRL_NET_DUPLICATE_IPPRIVATE_NETWORK_NAME 1512
PRL_NET_DUPLICATE_IPPRIVATE_NETWORK_NAME macro 1512
PRL_NET_DUPLICATE_VIRTUAL_NETWORK_ID 1512
PRL_NET_DUPLICATE_VIRTUAL_NETWORK_ID macro 1512
PRL_NET_ERR_ADAPTER_CONFIG 1512
PRL_NET_ERR_ADAPTER_CONFIG macro 1512
PRL_NET_ERR_ETH_NO_BINDABLE_ADAPTER 1513
PRL_NET_ERR_ETH_NO_BINDABLE_ADAPTER macro 1513
PRL_NET_ERR_PRL_NO_BINDABLE_ADAPTER 1513
PRL_NET_ERR_PRL_NO_BINDABLE_ADAPTER macro 1513
PRL_NET_ETHLIST_CREATE_ERROR 1513
PRL_NET_ETHLIST_CREATE_ERROR macro 1513
PRL_NET_INSTALL_FAILED 1513
PRL_NET_INSTALL_FAILED macro 1513
PRL_NET_INSTALL_TIMEOUT 1513
PRL_NET_INSTALL_TIMEOUT macro 1513
PRL_NET_IPADDRESS_MODIFY_FAILED_WARNING 1514
PRL_NET_IPADDRESS_MODIFY_FAILED_WARNING macro 1514
PRL_NET_IPPRIVATE_NETWORK_DOES_NOT_EXIST 1514

PRL_NET_IPPRIVATE_NETWORK_DOES_NOT_EXIST macro 1514

PRL_NET_IPPRIVATE_NETWORK_INVALID_IP 1514

PRL_NET_IPPRIVATE_NETWORK_INVALID_IP macro 1514

PRL_NET_OFFMGMT_ERROR 1514

PRL_NET_OFFMGMT_ERROR macro 1514

PRL_NET_PKTFILTER_ERROR 1515

PRL_NET_PKTFILTER_ERROR macro 1515

PRL_NET_PKTS_IN_PTRN 1515

PRL_NET_PKTS_IN_PTRN macro 1515

PRL_NET_PKTS_OUT_PTRN 1515

PRL_NET_PKTS_OUT_PTRN macro 1515

PRL_NET_PRLNET_OPEN_FAILED 1515

PRL_NET_PRLNET_OPEN_FAILED macro 1515

PRL_NET_RENAME_FAILED 1516

PRL_NET_RENAME_FAILED macro 1516

PRL_NET_RESTORE_DEFAULTS_PARTITIAL_SUCCESS 1516

PRL_NET_RESTORE_DEFAULTS_PARTITIAL_SUCCESS macro 1516

PRL_NET_SRV_NOTIFY_ERROR 1516

PRL_NET_SRV_NOTIFY_ERROR macro 1516

PRL_NET_SYSTEM_ERROR 1516

PRL_NET_SYSTEM_ERROR macro 1516

PRL_NET_UNINSTALL_FAILED 1516

PRL_NET_UNINSTALL_FAILED macro 1516

PRL_NET_VALID_DHCP_RANGE_WRONG_IP_ADDRS 1517

PRL_NET_VALID_DHCP_RANGE_WRONG_IP_ADDRS macro 1517

PRL_NET_VALID_DHCP_SCOPE_RANGE_LESS_MIN 1517

PRL_NET_VALID_DHCP_SCOPE_RANGE_LESS_MIN macro 1517

PRL_NET_VALID_FAILURE 1517

PRL_NET_VALID_FAILURE macro 1517

PRL_NET_VALID_MISMATCH_DHCP_SCOPE_MASK 1517

PRL_NET_VALID_MISMATCH_DHCP_SCOPE_MASK macro 1517

PRL_NET_VALID_WRONG_DHCP_IP_ADDR 1518

PRL_NET_VALID_WRONG_DHCP_IP_ADDR macro 1518

PRL_NET_VALID_WRONG_HOST_IP_ADDR 1518

PRL_NET_VALID_WRONG_HOST_IP_ADDR macro 1518

PRL_NET_VALID_WRONG_NET_MASK 1518

PRL_NET_VALID_WRONG_NET_MASK macro 1518

PRL_NET_VIRTUAL_NETWORK_ID_NOT_EXISTS 1518

PRL_NET_VIRTUAL_NETWORK_ID_NOT_EXISTS macro 1518

PRL_NET_VIRTUAL_NETWORK_NOT_FOUND 1519

PRL_NET_VIRTUAL_NETWORK_NOT_FOUND macro 1519

PRL_NET_VIRTUAL_NETWORK_SHARED_EXISTS 1519

PRL_NET_VIRTUAL_NETWORK_SHARED_EXISTS macro 1519

PRL_NET_VIRTUAL_NETWORK_SHARED_PROHIBITED 1519

PRL_NET_VIRTUAL_NETWORK_SHARED_PROHIBITED macro 1519

PRL_NET_VIRTUAL_NETWORK_TYPE 1081

PRL_NET_VIRTUAL_NETWORK_TYPE enumeration 1081

PRL_NET_VIRTUAL_NETWORK_TYPE_PTR 825

PRL_NET_VIRTUAL_NETWORK_TYPE_PTR type 825

PRL_NET_VLAN_UNSUPPORTED_IN_THIS_VERSION 1519

PRL_NET_VLAN_UNSUPPORTED_IN_THIS_VERSION macro 1519

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_CONFIG_ERRO R
1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_CONFIG_ERRO R
macro 1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_DISABLED 1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_DISABLED macro 1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NO_ADAPTER 1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NO_ADAPTER macro 1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NOT_EXIST 1520

PRL_NET_VMDEVICE_VIRTUAL_NETWORK_NOT_EXIST macro 1520

PRL_NET_WINSCM_OPEN_ERROR 1521

PRL_NET_WINSCM_OPEN_ERROR macro 1521

PRL_NETFILTER_MODE 1081

PRL_NETFILTER_MODE enumeration 1081

PRL_NETFILTER_MODE_PTR 825

PRL_NETFILTER_MODE_PTR type 825

PRL_NETWORK_RATE_MPU 1082
PRL_NETWORK_RATE_MPU enumeration 1082
PRL_NETWORK_RATE_MPU_PTR 825
PRL_NETWORK_RATE_MPU_PTR type 825
PRL_NULL 1631
PRL_NULL macro 1631
PRL_ONLINE_COMPACT_MODE 1082
PRL_ONLINE_COMPACT_MODE enumeration 1082
PRL_ONLINE_COMPACT_MODE_PTR 825
PRL_ONLINE_COMPACT_MODE_PTR type 825
PRL_OPTIMIZE_MODIFIERS_MODE 1082
PRL_OPTIMIZE_MODIFIERS_MODE enumeration 1082
PRL_OPTIMIZE_MODIFIERS_MODE_PTR 825
PRL_OPTIMIZE_MODIFIERS_MODE_PTR type 825
PRL_PARAM_FIELD_DATA_TYPE 1083
PRL_PARAM_FIELD_DATA_TYPE enumeration 1083
PRL_PARAM_FIELD_DATA_TYPE_PTR 826
PRL_PARAM_FIELD_DATA_TYPE_PTR type 826
PRL_PORT_FORWARDING_TYPE 1083
PRL_PORT_FORWARDING_TYPE enumeration 1083
PRL_PORT_FORWARDING_TYPE_PTR 826
PRL_PORT_FORWARDING_TYPE_PTR type 826
PRL_PRINTER_INTERFACE_TYPE 1083
PRL_PRINTER_INTERFACE_TYPE enumeration 1083
PRL_PRINTER_INTERFACE_TYPE_PTR 826
PRL_PRINTER_INTERFACE_TYPE_PTR type 826
PRL_PRIVATE_EVENT_TYPE 1084
PRL_PRIVATE_EVENT_TYPE enumeration 1084
PRL_PRIVATE_EVENT_TYPE_PTR 826
PRL_PRIVATE_EVENT_TYPE_PTR type 826
PRL_PRIVILEGED_GUEST_OS_SESSION 1632
PRL_PRIVILEGED_GUEST_OS_SESSION macro 1632
PRL_PROBLEM_REPORT_FLAGS 1086
PRL_PROBLEM_REPORT_FLAGS enumeration 1086
PRL_PROBLEM_REPORT_FLAGS_PTR 826
PRL_PROBLEM_REPORT_FLAGS_PTR type 826
PRL_PROBLEM_REPORT_REASON 1086
PRL_PROBLEM_REPORT_REASON enumeration 1086
PRL_PROBLEM_REPORT_REASON_PTR 827
PRL_PROBLEM_REPORT_REASON_PTR type 827
PRL_PROBLEM_REPORT_SCHEME 1087
PRL_PROBLEM_REPORT_SCHEME enumeration 1087
PRL_PROBLEM_REPORT_SCHEME_PTR 827
PRL_PROBLEM_REPORT_SCHEME_PTR type 827
PRL_PROBLEM_REPORT_TYPE 1087
PRL_PROBLEM_REPORT_TYPE enumeration 1087
PRL_PROBLEM_REPORT_TYPE_PTR 827
PRL_PROBLEM_REPORT_TYPE_PTR type 827
PRL_PROCESS_STATE_TYPE 1088
PRL_PROCESS_STATE_TYPE enumeration 1088
PRL_PROCESS_STATE_TYPE_PTR 827
PRL_PROCESS_STATE_TYPE_PTR type 827
PRL_PRODUCT_ANY enumeration member 1021
PRL_PRODUCT_DESKTOP enumeration member 1021
PRL_PRODUCT_DESKTOP_WL enumeration member 1021
PRL_PRODUCT_ENUM 1088
PRL_PRODUCT_ENUM enumeration 1088
PRL_PRODUCT_ENUM_PTR 827
PRL_PRODUCT_ENUM_PTR type 827
PRL_PRODUCT_PLAYER enumeration member 1021
PRL_PRODUCT_SERVER enumeration member 1021
PRL_PRODUCT_STM enumeration member 1021
PRL_PRODUCT_WORKSTATION enumeration member 1021
PRL_PROXY_CONNECTION_STATUS 1089
PRL_PROXY_CONNECTION_STATUS enumeration 1089
PRL_PROXY_CONNECTION_STATUS_PTR 828
PRL_PROXY_CONNECTION_STATUS_PTR type 828
PRL_PROXY_MAX_HOSTS 1632
PRL_PROXY_MAX_HOSTS macro 1632
PRL_QUERY_ELEMENT_FLAGS 1089
PRL_QUERY_ELEMENT_FLAGS enumeration 1089
PRL_QUERY_ELEMENT_FLAGS_PTR 828
PRL_QUERY_ELEMENT_FLAGS_PTR type 828
PRL_QUEST Macros 1521
PRL_QUEST_CD_DRIVE_NOT_EXIST 1521
PRL_QUEST_CD_DRIVE_NOT_EXIST macro 1521
PRL_QUEST_CDD_IMAGE_NOT_EXIST 1522
PRL_QUEST_CDD_IMAGE_NOT_EXIST macro 1522

PRL_QUEST_EXCEED_RECOMMENDED_MEMORY 1522
PRL_QUEST_EXCEED_RECOMMENDED_MEMORY macro 1522
PRL_QUEST_FDD_DRIVE_NOT_EXIST 1522
PRL_QUEST_FDD_DRIVE_NOT_EXIST macro 1522
PRL_QUEST_FDD_IMAGE_NOT_EXIST 1522
PRL_QUEST_FDD_IMAGE_NOT_EXIST macro 1522
PRL_QUEST_HDD_IMAGE_NOT_EXIST 1522
PRL_QUEST_HDD_IMAGE_NOT_EXIST macro 1522
PRL_QUEST_HDD_SAME_IMAGE 1523
PRL_QUEST_HDD_SAME_IMAGE macro 1523
PRL_QUEST_POWER_OFF 1523
PRL_QUEST_POWER_OFF macro 1523
PRL_QUEST_RESET 1523
PRL_QUEST_RESET macro 1523
PRL_QUESTION Macros 1523
PRL_QUESTION_ASK_ENCRYPTED_VM_PASSWORD 1524
PRL_QUESTION_ASK_ENCRYPTED_VM_PASSWORD macro 1524
PRL_QUESTION_CAN_NOT_GET_DISK_FREE_SPACE 1524
PRL_QUESTION_CAN_NOT_GET_DISK_FREE_SPACE macro 1524
PRL_QUESTION_CONVERT_3RD_PARTY_CANNOT_MIGRATE 1524
PRL_QUESTION_CONVERT_3RD_PARTY_CANNOT_MIGRATE macro 1524
PRL_QUESTION_CONVERT_GUEST_OS_IS_HIBERNATED 1524
PRL_QUESTION_CONVERT_GUEST_OS_IS_HIBERNATED macro 1524
PRL_QUESTION_CONVERT_VM_CANT_DETECT_OS 1525
PRL_QUESTION_CONVERT_VM_CANT_DETECT_OS macro 1525
PRL_QUESTION_CONVERT_VM_SPECIFY_DISTRO_PATH 1525
PRL_QUESTION_CONVERT_VM_SPECIFY_DISTRO_PATH macro 1525
PRL_QUESTION_FINALIZE_VM_PROCESS 1525
PRL_QUESTION_FINALIZE_VM_PROCESS macro 1525
PRL_QUESTION_SETUP_YANDEX_SEARCH 1525
PRL_QUESTION_SETUP_YANDEX_SEARCH macro 1525
PRL_REGISTER_3PARTY_VM_FLAGS 1089
PRL_REGISTER_3PARTY_VM_FLAGS enumeration 1089
PRL_REGISTER_3PARTY_VM_FLAGS_PTR 828
PRL_REGISTER_3PARTY_VM_FLAGS_PTR type 828
PRL_REGISTER_NEW_VM 1089
PRL_REGISTER_NEW_VM enumeration 1089
PRL_REGISTER_NEW_VM_PTR 828
PRL_REGISTER_NEW_VM_PTR type 828
PRL_REGISTER_VM_FLAGS 1090
PRL_REGISTER_VM_FLAGS enumeration 1090
PRL_REGISTER_VM_FLAGS_PTR 828
PRL_REGISTER_VM_FLAGS_PTR type 828
PRL_RESIZE_IMAGE_FLAGS 1090
PRL_RESIZE_IMAGE_FLAGS enumeration 1090
PRL_RESIZE_IMAGE_FLAGS_PTR 829
PRL_RESIZE_IMAGE_FLAGS_PTR type 829
PRL_RESULT_DECLARE_ERROR 1632
PRL_RESULT_DECLARE_ERROR macro 1632
PRL_RESULT_DECLARE_SUCCESS 1632
PRL_RESULT_DECLARE_SUCCESS macro 1632
prl_result_to_string 745
prl_result_to_string function 745
PRL_RUNNING_TASK_TYPE_ENUM 1090
PRL_RUNNING_TASK_TYPE_ENUM enumeration 1090
PRL_RUNNING_TASK_TYPE_ENUM_PTR 829
PRL_RUNNING_TASK_TYPE_ENUM_PTR type 829
PRL_SATA_READ_REQUESTS_PTRN 1633
PRL_SATA_READ_REQUESTS_PTRN macro 1633
PRL_SATA_READ_TOTAL_PTRN 1633
PRL_SATA_READ_TOTAL_PTRN macro 1633
PRL_SATA_WRITE_REQUESTS_PTRN 1633
PRL_SATA_WRITE_REQUESTS_PTRN macro 1633
PRL_SATA_WRITE_TOTAL_PTRN 1633
PRL_SATA_WRITE_TOTAL_PTRN macro 1633
PRL_SCALE_VIEW_MODE 1091
PRL_SCALE_VIEW_MODE enumeration 1091
PRL_SCALE_VIEW_MODE_PTR 829
PRL_SCALE_VIEW_MODE_PTR type 829
PRL_SCAN_CODE Macros 1526
PRL_SCAN_CODE_0 1529

PRL_SCAN_CODE_0 macro 1529
PRL_SCAN_CODE_1 1530
PRL_SCAN_CODE_1 macro 1530
PRL_SCAN_CODE_2 1530
PRL_SCAN_CODE_2 macro 1530
PRL_SCAN_CODE_3 1530
PRL_SCAN_CODE_3 macro 1530
PRL_SCAN_CODE_4 1530
PRL_SCAN_CODE_4 macro 1530
PRL_SCAN_CODE_5 1530
PRL_SCAN_CODE_5 macro 1530
PRL_SCAN_CODE_6 1531
PRL_SCAN_CODE_6 macro 1531
PRL_SCAN_CODE_7 1531
PRL_SCAN_CODE_7 macro 1531
PRL_SCAN_CODE_8 1531
PRL_SCAN_CODE_8 macro 1531
PRL_SCAN_CODE_9 1531
PRL_SCAN_CODE_9 macro 1531
PRL_SCAN_CODE_A 1531
PRL_SCAN_CODE_A macro 1531
PRL_SCAN_CODE_ALT_LEFT 1532
PRL_SCAN_CODE_ALT_LEFT macro 1532
PRL_SCAN_CODE_ALT_RIGHT 1532
PRL_SCAN_CODE_ALT_RIGHT macro 1532
PRL_SCAN_CODE_APP_CALCULATOR 1532
PRL_SCAN_CODE_APP_CALCULATOR macro 1532
PRL_SCAN_CODE_APP_MAIL 1532
PRL_SCAN_CODE_APP_MAIL macro 1532
PRL_SCAN_CODE_APP_MY_COMPUTER 1532
PRL_SCAN_CODE_APP_MY_COMPUTER macro 1532
PRL_SCAN_CODE_B 1533
PRL_SCAN_CODE_B macro 1533
PRL_SCAN_CODE_BACKSLASH 1533
PRL_SCAN_CODE_BACKSLASH macro 1533
PRL_SCAN_CODE_BACKSPACE 1533
PRL_SCAN_CODE_BACKSPACE macro 1533
PRL_SCAN_CODE_BRAZILIAN_KEYPAD 1533
PRL_SCAN_CODE_BRAZILIAN_KEYPAD macro 1533
PRL_SCAN_CODE_BREAK 1533
PRL_SCAN_CODE_BREAK macro 1533
PRL_SCAN_CODE_C 1534
PRL_SCAN_CODE_C macro 1534
PRL_SCAN_CODE_CAPS_LOCK 1534
PRL_SCAN_CODE_CAPS_LOCK macro 1534
PRL_SCAN_CODE_CBRACE_LEFT 1534
PRL_SCAN_CODE_CBRACE_LEFT macro 1534
PRL_SCAN_CODE_CBRACE_RIGHT 1534
PRL_SCAN_CODE_CBRACE_RIGHT macro 1534
PRL_SCAN_CODE_CMD_LEFT 1534
PRL_SCAN_CODE_CMD_LEFT macro 1534
PRL_SCAN_CODE_CMD_RIGHT 1535
PRL_SCAN_CODE_CMD_RIGHT macro 1535
PRL_SCAN_CODE_COLON 1535
PRL_SCAN_CODE_COLON macro 1535
PRL_SCAN_CODE_CTRL_LEFT 1535
PRL_SCAN_CODE_CTRL_LEFT macro 1535
PRL_SCAN_CODE_CTRL_RIGHT 1535
PRL_SCAN_CODE_CTRL_RIGHT macro 1535
PRL_SCAN_CODE_D 1535
PRL_SCAN_CODE_D macro 1535
PRL_SCAN_CODE_DELETE 1536
PRL_SCAN_CODE_DELETE macro 1536
PRL_SCAN_CODE_DOLLAR 1536
PRL_SCAN_CODE_DOLLAR macro 1536
PRL_SCAN_CODE_DOWN 1536
PRL_SCAN_CODE_DOWN macro 1536
PRL_SCAN_CODE_E 1536
PRL_SCAN_CODE_E macro 1536
PRL_SCAN_CODE_EJECT 1536
PRL_SCAN_CODE_EJECT macro 1536
PRL_SCAN_CODE_END 1537
PRL_SCAN_CODE_END macro 1537
PRL_SCAN_CODE_ENTER 1537
PRL_SCAN_CODE_ENTER macro 1537
PRL_SCAN_CODE_ESC 1537
PRL_SCAN_CODE_ESC macro 1537
PRL_SCAN_CODE_EURO 1537

PRL_SCAN_CODE_EURO macro 1537
PRL_SCAN_CODE_EUROPE_1 1538
PRL_SCAN_CODE_EUROPE_1 macro 1538
PRL_SCAN_CODE_F 1538
PRL_SCAN_CODE_F macro 1538
PRL_SCAN_CODE_F1 1538
PRL_SCAN_CODE_F1 macro 1538
PRL_SCAN_CODE_F10 1538
PRL_SCAN_CODE_F10 macro 1538
PRL_SCAN_CODE_F11 1538
PRL_SCAN_CODE_F11 macro 1538
PRL_SCAN_CODE_F12 1539
PRL_SCAN_CODE_F12 macro 1539
PRL_SCAN_CODE_F13 1539
PRL_SCAN_CODE_F13 macro 1539
PRL_SCAN_CODE_F14 1539
PRL_SCAN_CODE_F14 macro 1539
PRL_SCAN_CODE_F15 1539
PRL_SCAN_CODE_F15 macro 1539
PRL_SCAN_CODE_F16 1539
PRL_SCAN_CODE_F16 macro 1539
PRL_SCAN_CODE_F17 1540
PRL_SCAN_CODE_F17 macro 1540
PRL_SCAN_CODE_F18 1540
PRL_SCAN_CODE_F18 macro 1540
PRL_SCAN_CODE_F19 1540
PRL_SCAN_CODE_F19 macro 1540
PRL_SCAN_CODE_F2 1540
PRL_SCAN_CODE_F2 macro 1540
PRL_SCAN_CODE_F20 1540
PRL_SCAN_CODE_F20 macro 1540
PRL_SCAN_CODE_F21 1541
PRL_SCAN_CODE_F21 macro 1541
PRL_SCAN_CODE_F22 1541
PRL_SCAN_CODE_F22 macro 1541
PRL_SCAN_CODE_F23 1541
PRL_SCAN_CODE_F23 macro 1541
PRL_SCAN_CODE_F24 1541
PRL_SCAN_CODE_F24 macro 1541
PRL_SCAN_CODE_F3 1541
PRL_SCAN_CODE_F3 macro 1541
PRL_SCAN_CODE_F4 1542
PRL_SCAN_CODE_F4 macro 1542
PRL_SCAN_CODE_F5 1542
PRL_SCAN_CODE_F5 macro 1542
PRL_SCAN_CODE_F6 1542
PRL_SCAN_CODE_F6 macro 1542
PRL_SCAN_CODE_F7 1542
PRL_SCAN_CODE_F7 macro 1542
PRL_SCAN_CODE_F8 1542
PRL_SCAN_CODE_F8 macro 1542
PRL_SCAN_CODE_F9 1543
PRL_SCAN_CODE_F9 macro 1543
PRL_SCAN_CODE_G 1543
PRL_SCAN_CODE_G macro 1543
PRL_SCAN_CODE_GREATER 1543
PRL_SCAN_CODE_GREATER macro 1543
PRL_SCAN_CODE_H 1543
PRL_SCAN_CODE_H macro 1543
PRL_SCAN_CODE_HANGUEL 1543
PRL_SCAN_CODE_HANGUEL macro 1543
PRL_SCAN_CODE_HANJA 1544
PRL_SCAN_CODE_HANJA macro 1544
PRL_SCAN_CODE_HENKAN 1544
PRL_SCAN_CODE_HENKAN macro 1544
PRL_SCAN_CODE_HIRAGANA 1544
PRL_SCAN_CODE_HIRAGANA macro 1544
PRL_SCAN_CODE_HIRAGANA_KATAKANA 1544
PRL_SCAN_CODE_HIRAGANA_KATAKANA macro 1544
PRL_SCAN_CODE_HOME 1544
PRL_SCAN_CODE_HOME macro 1544
PRL_SCAN_CODE_I 1545
PRL_SCAN_CODE_I macro 1545
PRL_SCAN_CODE_INSERT 1545
PRL_SCAN_CODE_INSERT macro 1545
PRL_SCAN_CODE_J 1545
PRL_SCAN_CODE_J macro 1545
PRL_SCAN_CODE_K 1545

PRL_SCAN_CODE_K macro 1545
PRL_SCAN_CODE_KATAKANA 1545
PRL_SCAN_CODE_KATAKANA macro 1545
PRL_SCAN_CODE_L 1546
PRL_SCAN_CODE_L macro 1546
PRL_SCAN_CODE_LEFT 1546
PRL_SCAN_CODE_LEFT macro 1546
PRL_SCAN_CODE_LESS 1546
PRL_SCAN_CODE_LESS macro 1546
PRL_SCAN_CODE_LESS_GREATER 1546
PRL_SCAN_CODE_LESS_GREATER macro 1546
PRL_SCAN_CODE_M 1546
PRL_SCAN_CODE_M macro 1546
PRL_SCAN_CODE_MEDIA_NEXT_TRACK 1547
PRL_SCAN_CODE_MEDIA_NEXT_TRACK macro 1547
PRL_SCAN_CODE_MEDIA_PLAY_PAUSE 1547
PRL_SCAN_CODE_MEDIA_PLAY_PAUSE macro 1547
PRL_SCAN_CODE_MEDIA_PREV_TRACK 1547
PRL_SCAN_CODE_MEDIA_PREV_TRACK macro 1547
PRL_SCAN_CODE_MEDIA_SELECT 1547
PRL_SCAN_CODE_MEDIA_SELECT macro 1547
PRL_SCAN_CODE_MEDIA_STOP 1547
PRL_SCAN_CODE_MEDIA_STOP macro 1547
PRL_SCAN_CODE_MENU 1548
PRL_SCAN_CODE_MENU macro 1548
PRL_SCAN_CODE_MINUS 1548
PRL_SCAN_CODE_MINUS macro 1548
PRL_SCAN_CODE_MUHENKAN 1548
PRL_SCAN_CODE_MUHENKAN macro 1548
PRL_SCAN_CODE_MUTE 1548
PRL_SCAN_CODE_MUTE macro 1548
PRL_SCAN_CODE_N 1548
PRL_SCAN_CODE_N macro 1548
PRL_SCAN_CODE_NUM_LOCK 1549
PRL_SCAN_CODE_NUM_LOCK macro 1549
PRL_SCAN_CODE_O 1549
PRL_SCAN_CODE_O macro 1549
PRL_SCAN_CODE_P 1549
PRL_SCAN_CODE_P macro 1549
PRL_SCAN_CODE_PAD_0 1549
PRL_SCAN_CODE_PAD_0 macro 1549
PRL_SCAN_CODE_PAD_1 1549
PRL_SCAN_CODE_PAD_1 macro 1549
PRL_SCAN_CODE_PAD_2 1550
PRL_SCAN_CODE_PAD_2 macro 1550
PRL_SCAN_CODE_PAD_3 1550
PRL_SCAN_CODE_PAD_3 macro 1550
PRL_SCAN_CODE_PAD_4 1550
PRL_SCAN_CODE_PAD_4 macro 1550
PRL_SCAN_CODE_PAD_5 1550
PRL_SCAN_CODE_PAD_5 macro 1550
PRL_SCAN_CODE_PAD_6 1550
PRL_SCAN_CODE_PAD_6 macro 1550
PRL_SCAN_CODE_PAD_7 1551
PRL_SCAN_CODE_PAD_7 macro 1551
PRL_SCAN_CODE_PAD_8 1551
PRL_SCAN_CODE_PAD_8 macro 1551
PRL_SCAN_CODE_PAD_9 1551
PRL_SCAN_CODE_PAD_9 macro 1551
PRL_SCAN_CODE_PAD_DEL 1551
PRL_SCAN_CODE_PAD_DEL macro 1551
PRL_SCAN_CODE_PAD_ENTER 1551
PRL_SCAN_CODE_PAD_ENTER macro 1551
PRL_SCAN_CODE_PAD_EQUAL 1552
PRL_SCAN_CODE_PAD_EQUAL macro 1552
PRL_SCAN_CODE_PAD_MINUS 1552
PRL_SCAN_CODE_PAD_MINUS macro 1552
PRL_SCAN_CODE_PAD_PLUS 1552
PRL_SCAN_CODE_PAD_PLUS macro 1552
PRL_SCAN_CODE_PAD_SLASH 1552
PRL_SCAN_CODE_PAD_SLASH macro 1552
PRL_SCAN_CODE_PAD_STAR 1552
PRL_SCAN_CODE_PAD_STAR macro 1552
PRL_SCAN_CODE_PAGE_DOWN 1553
PRL_SCAN_CODE_PAGE_DOWN macro 1553
PRL_SCAN_CODE_PAGE_UP 1553
PRL_SCAN_CODE_PAGE_UP macro 1553
PRL_SCAN_CODE_PAUSE_BREAK 1553

PRL_SCAN_CODE_PAUSE_BREAK macro 1553
PRL_SCAN_CODE_PC9800_KEYPAD 1553
PRL_SCAN_CODE_PC9800_KEYPAD macro 1553
PRL_SCAN_CODE_PLUS 1553
PRL_SCAN_CODE_PLUS macro 1553
PRL_SCAN_CODE_PRINT_SCREEN 1554
PRL_SCAN_CODE_PRINT_SCREEN macro 1554
PRL_SCAN_CODE_PRINT_SCREEN2 1554
PRL_SCAN_CODE_PRINT_SCREEN2 macro 1554
PRL_SCAN_CODE_Q 1554
PRL_SCAN_CODE_Q macro 1554
PRL_SCAN_CODE_QUOTE 1554
PRL_SCAN_CODE_QUOTE macro 1554
PRL_SCAN_CODE_R 1554
PRL_SCAN_CODE_R macro 1554
PRL_SCAN_CODE_RIGHT 1555
PRL_SCAN_CODE_RIGHT macro 1555
PRL_SCAN_CODE_RO 1555
PRL_SCAN_CODE_RO macro 1555
PRL_SCAN_CODE_S 1555
PRL_SCAN_CODE_S macro 1555
PRL_SCAN_CODE_SCROLL_LOCK 1555
PRL_SCAN_CODE_SCROLL_LOCK macro 1555
PRL_SCAN_CODE_SHIFT_LEFT 1555
PRL_SCAN_CODE_SHIFT_LEFT macro 1555
PRL_SCAN_CODE_SHIFT_RIGHT 1556
PRL_SCAN_CODE_SHIFT_RIGHT macro 1556
PRL_SCAN_CODE_SLASH 1556
PRL_SCAN_CODE_SLASH macro 1556
PRL_SCAN_CODE_SPACE 1556
PRL_SCAN_CODE_SPACE macro 1556
PRL_SCAN_CODE_SYSRQ 1556
PRL_SCAN_CODE_SYSRQ macro 1556
PRL_SCAN_CODE_SYSTEM_POWER 1556
PRL_SCAN_CODE_SYSTEM_POWER macro 1556
PRL_SCAN_CODE_SYSTEM_SLEEP 1557
PRL_SCAN_CODE_SYSTEM_SLEEP macro 1557
PRL_SCAN_CODE_SYSTEM_WAKE 1557
PRL_SCAN_CODE_SYSTEM_WAKE macro 1557
PRL_SCAN_CODE_T 1557
PRL_SCAN_CODE_T macro 1557
PRL_SCAN_CODE_TAB 1557
PRL_SCAN_CODE_TAB macro 1557
PRL_SCAN_CODE_TILDA 1557
PRL_SCAN_CODE_TILDA macro 1557
PRL_SCAN_CODE_U 1558
PRL_SCAN_CODE_U macro 1558
PRL_SCAN_CODE_UP 1558
PRL_SCAN_CODE_UP macro 1558
PRL_SCAN_CODE_V 1558
PRL_SCAN_CODE_V macro 1558
PRL_SCAN_CODE_VOLUME_DOWN 1558
PRL_SCAN_CODE_VOLUME_DOWN macro 1558
PRL_SCAN_CODE_VOLUME_UP 1558
PRL_SCAN_CODE_VOLUME_UP macro 1558
PRL_SCAN_CODE_W 1559
PRL_SCAN_CODE_W macro 1559
PRL_SCAN_CODE_WWW_BACK 1559
PRL_SCAN_CODE_WWW_BACK macro 1559
PRL_SCAN_CODE_WWW_FAVORITES 1559
PRL_SCAN_CODE_WWW_FAVORITES macro 1559
PRL_SCAN_CODE_WWW_FORWARD 1559
PRL_SCAN_CODE_WWW_FORWARD macro 1559
PRL_SCAN_CODE_WWW_HOME 1559
PRL_SCAN_CODE_WWW_HOME macro 1559
PRL_SCAN_CODE_WWW_REFRESH 1560
PRL_SCAN_CODE_WWW_REFRESH macro 1560
PRL_SCAN_CODE_WWW_SEARCH 1560
PRL_SCAN_CODE_WWW_SEARCH macro 1560
PRL_SCAN_CODE_WWW_STOP 1560
PRL_SCAN_CODE_WWW_STOP macro 1560
PRL_SCAN_CODE_X 1560
PRL_SCAN_CODE_X macro 1560
PRL_SCAN_CODE_Y 1560
PRL_SCAN_CODE_Y macro 1560
PRL_SCAN_CODE_YEN 1561
PRL_SCAN_CODE_YEN macro 1561
PRL_SCAN_CODE_Z 1561

PRL_SCAN_CODE_Z macro 1561
PRL_SCAN_CODE_ZENKAKU_HANKAKU 1561
PRL_SCAN_CODE_ZENKAKU_HANKAKU macro 1561
PRL_SCSI_READ_REQUESTS_PTRN 1634
PRL_SCSI_READ_REQUESTS_PTRN macro 1634
PRL_SCSI_READ_TOTAL_PTRN 1634
PRL_SCSI_READ_TOTAL_PTRN macro 1634
PRL_SCSI_WRITE_REQUESTS_PTRN 1634
PRL_SCSI_WRITE_REQUESTS_PTRN macro 1634
PRL_SCSI_WRITE_TOTAL_PTRN 1634
PRL_SCSI_WRITE_TOTAL_PTRN macro 1634
PRL_SDK_WRAP_FOR_EACH 1635
PRL_SDK_WRAP_FOR_EACH macro 1635
PRL_SECURITY_LEVEL 1091
PRL_SECURITY_LEVEL enumeration 1091
PRL_SECURITY_LEVEL_PTR 829
PRL_SECURITY_LEVEL_PTR type 829
PRL_SERIAL_PORT_SOCKET_OPERATION_MODE 1092
PRL_SERIAL_PORT_SOCKET_OPERATION_MODE enumeration 1092
PRL_SERIAL_PORT_SOCKET_OPERATION_MODE_PTR 829
PRL_SERIAL_PORT_SOCKET_OPERATION_MODE_PTR type 829
PRL_SERVICE_STATUS_ENUM 1092
PRL_SERVICE_STATUS_ENUM enumeration 1092
PRL_SERVICE_STATUS_ENUM_PTR 830
PRL_SERVICE_STATUS_ENUM_PTR type 830
PRL_SET_USERPASSWD_FLAGS 1092
PRL_SET_USERPASSWD_FLAGS enumeration 1092
PRL_SET_USERPASSWD_FLAGS_PTR 830
PRL_SET_USERPASSWD_FLAGS_PTR type 830
PRL_SIZE 830
PRL_SIZE type 830
PRL_SIZE_PTR 830
PRL_SIZE_PTR type 830
PRL_SIZE_PTR_PTR 831
PRL_SIZE_PTR_PTR type 831
PRL_SNAPSHOT_ELEMENT 1093
PRL_SNAPSHOT_ELEMENT structure 1093
PRL_SNAPSHOT_ELEMENT_PTR 831
PRL_SNAPSHOT_ELEMENT_PTR type 831
PRL_SOUND_CHANNEL_TYPE 1093
PRL_SOUND_CHANNEL_TYPE enumeration 1093
PRL_SOUND_CHANNEL_TYPE_PTR 831
PRL_SOUND_CHANNEL_TYPE_PTR type 831
PRL_SRV_SHUTDOWN_FLAGS 1093
PRL_SRV_SHUTDOWN_FLAGS enumeration 1093
PRL_SRV_SHUTDOWN_FLAGS_PTR 831
PRL_SRV_SHUTDOWN_FLAGS_PTR type 831
PRL_STATE Macros 1561
PRL_STATE_CALLBACK_INVALID 1562
PRL_STATE_CALLBACK_INVALID macro 1562
PRL_STATE_CORE_FIRST_COMPLETE_CALLBACK 1562
PRL_STATE_CORE_FIRST_COMPLETE_CALLBACK macro 1562
PRL_STATE_CORE_FULL_COMPLETE_CALLBACK 1562
PRL_STATE_CORE_FULL_COMPLETE_CALLBACK macro 1562
PRL_STATE_CORE_START_DISK_PROCESS_CALLBACK 1562
PRL_STATE_CORE_START_DISK_PROCESS_CALLBACK macro 1562
PRL_STATE_DISK_CALLBACK 1563
PRL_STATE_DISK_CALLBACK macro 1563
PRL_STATE_DISK_CALLBACK_STAGE_CPL 1563
PRL_STATE_DISK_CALLBACK_STAGE_CPL macro 1563
PRL_STATE_PROGRESS_COMPLETED 1563
PRL_STATE_PROGRESS_COMPLETED macro 1563
PRL_STATE_PROGRESS_MAX 1563
PRL_STATE_PROGRESS_MAX macro 1563
PRL_STATE_PROGRESS_MIN 1563
PRL_STATE_PROGRESS_MIN macro 1563
PRL_STATES_CALLBACK 780
PRL_STATES_CALLBACK type 780
PRL_STATES_CALLBACK_TYPE 831
PRL_STATES_CALLBACK_TYPE type 831
PRL_STATES_CALLBACK_TYPE_PTR 832
PRL_STATES_CALLBACK_TYPE_PTR type 832
PRL_STR 832
PRL_STR type 832
PRL_STRUCT 1651

PRL_STRUCT macro 1651
PRL_SUCCEEDED 1651
PRL_SUCCEEDED macro 1651
PRL_SWITCH_SNAPSHOT_FLAGS 1094
PRL_SWITCH_SNAPSHOT_FLAGS enumeration 1094
PRL_TASKBAR_STATES 1094
PRL_TASKBAR_STATES enumeration 1094
PRL_TERNARY_FALSE 1652
PRL_TERNARY_FALSE macro 1652
PRL_TERNARY_TRUE 1652
PRL_TERNARY_TRUE macro 1652
PRL_TERNARY_UNDEF 1652
PRL_TERNARY_UNDEF macro 1652
PRL_TIS Types 796
PRL_TIS_RECORD_FIELD 1094
PRL_TIS_RECORD_FIELD enumeration 1094
PRL_TIS_RECORD_FIELD_PTR 796
PRL_TIS_RECORD_FIELD_PTR type 796
PRL_TIS_RECORD_STATE 1095
PRL_TIS_RECORD_STATE enumeration 1095
PRL_TIS_RECORD_STATE_PTR 796
PRL_TIS_RECORD_STATE_PTR type 796
PRL_TOOLS Types 796
PRL_TOOLS_SIA_OPEN_IN_TYPES 1095
PRL_TOOLS_SIA_OPEN_IN_TYPES enumeration 1095
PRL_TOOLS_SIA_OPEN_IN_TYPES_PTR 797
PRL_TOOLS_SIA_OPEN_IN_TYPES_PTR type 797
PRL_TOOLS_UTILITY_COMMAND_FLAG 1095
PRL_TOOLS_UTILITY_COMMAND_FLAG enumeration 1095
PRL_TOOLS_UTILITY_COMMAND_FLAG_PTR 797
PRL_TOOLS_UTILITY_COMMAND_FLAG_PTR type 797
PRL_TOOLS_UTILITY_COMMAND_ID 1096
PRL_TOOLS_UTILITY_COMMAND_ID enumeration 1096
PRL_TOOLS_UTILITY_COMMAND_ID_PTR 797
PRL_TOOLS_UTILITY_COMMAND_ID_PTR type 797
PRL_TRUE 1652
PRL_TRUE macro 1652
PRL_UIEMU_ELEMENT_SEARCH_AREA_GLOBAL enumeration member 1021
PRL_UIEMU_ELEMENT_SEARCH_AREA_WINDOW enumeration member 1021
PRL_UINT Types 797
PRL_UINT16 798
PRL_UINT16 type 798
PRL_UINT16_PTR 798
PRL_UINT16_PTR type 798
PRL_UINT16_PTR_PTR 798
PRL_UINT16_PTR_PTR type 798
PRL_UINT32 798
PRL_UINT32 type 798
PRL_UINT32_PTR 799
PRL_UINT32_PTR type 799
PRL_UINT32_PTR_PTR 799
PRL_UINT32_PTR_PTR type 799
PRL_UINT64 799
PRL_UINT64 type 799
PRL_UINT64_PTR 799
PRL_UINT64_PTR type 799
PRL_UINT64_PTR_PTR 799
PRL_UINT64_PTR_PTR type 799
PRL_UINT8 800
PRL_UINT8 type 800
PRL_UINT8_PTR 800
PRL_UINT8_PTR type 800
PRL_UINT8_PTR_PTR 800
PRL_UINT8_PTR_PTR type 800
PRL_UNDO_DISKS_MODE 1096
PRL_UNDO_DISKS_MODE enumeration 1096
PRL_UNDO_DISKS_MODE_PTR 832
PRL_UNDO_DISKS_MODE_PTR type 832
PRL_UPDATE_LICENSE_FLAGS 1097
PRL_UPDATE_LICENSE_FLAGS enumeration 1097
PRL_UPDATE_LICENSE_FLAGS_PTR 832
PRL_UPDATE_LICENSE_FLAGS_PTR type 832
PRL_USB_DEVICE_AUTO_CONNECT_OPTION 1097
PRL_USB_DEVICE_AUTO_CONNECT_OPTION enumeration 1097
PRL_USB_DEVICE_AUTO_CONNECT_OPTION_PTR 833
PRL_USB_DEVICE_AUTO_CONNECT_OPTION_PTR type 833

PRL_USB_DEVICE_CONNECT_REASON 1097
PRL_USB_DEVICE_CONNECT_REASON enumeration 1097
PRL_USB_DEVICE_CONNECT_REASON_PTR 833
PRL_USB_DEVICE_CONNECT_REASON_PTR type 833
PRL_USB_DEVICE_TYPE 1098
PRL_USB_DEVICE_TYPE enumeration 1098
PRL_USB_DEVICE_TYPE_PTR 833
PRL_USB_DEVICE_TYPE_PTR type 833
PRL_USE_VNET_NAME_FOR_BRIDGE_NAME enumeration member 1033
PRL_VCMMD_COMMAND_FLAGS 1099
PRL_VCMMD_COMMAND_FLAGS enumeration 1099
PRL_VCMMD_COMMAND_FLAGS_PTR 833
PRL_VCMMD_COMMAND_FLAGS_PTR type 833
PRL_VERSION_10X enumeration member 1032
PRL_VERSION_3X enumeration member 1032
PRL_VERSION_4X enumeration member 1032
PRL_VERSION_5X enumeration member 1032
PRL_VERSION_6X enumeration member 1032
PRL_VERSION_7X enumeration member 1032
PRL_VERSION_8X enumeration member 1032
PRL_VERSION_9X enumeration member 1032
PRL_VERSION_ANY enumeration member 1032
PRL_VERSION_ENUM 1099
PRL_VERSION_ENUM enumeration 1099
PRL_VERSION_ENUM_PTR 833
PRL_VERSION_ENUM_PTR type 833
PRL_VIDEO_3D_ACCELERATION 1100
PRL_VIDEO_3D_ACCELERATION enumeration 1100
PRL_VIDEO_3D_ACCELERATION_PTR 834
PRL_VIDEO_3D_ACCELERATION_PTR type 834
PRL_VIDEO_RECEIVER_CONNECTION_STATE 1100
PRL_VIDEO_RECEIVER_CONNECTION_STATE enumeration 1100
PRL_VIDEO_RECEIVER_CONNECTION_STATE_PTR 834
PRL_VIDEO_RECEIVER_CONNECTION_STATE_PTR type 834
PRL_VIRTUAL_MACHINE_PROFILE 1100
PRL_VIRTUAL_MACHINE_PROFILE enumeration 1100
PRL_VIRTUAL_MACHINE_PROFILE_PTR 834
PRL_VIRTUAL_MACHINE_PROFILE_PTR type 834
PRL_VIRTUAL_NETWORK_FLAGS 1101
PRL_VIRTUAL_NETWORK_FLAGS enumeration 1101
PRL_VIRTUAL_NETWORK_FLAGS_PTR 834
PRL_VIRTUAL_NETWORK_FLAGS_PTR type 834
PRL_VM Structs 920
PRL_VM Types 800
PRL_VM_ACCELERATION_LEVEL 938
PRL_VM_ACCELERATION_LEVEL enumeration 938
PRL_VM_ACCELERATION_LEVEL_PTR 801
PRL_VM_ACCELERATION_LEVEL_PTR type 801
PRL_VM_ACCESS_FOR_OTHERS 938
PRL_VM_ACCESS_FOR_OTHERS enumeration 938
PRL_VM_ACCESS_FOR_OTHERS_PTR 802
PRL_VM_ACCESS_FOR_OTHERS_PTR type 802
PRL_VM_ACTION 938
PRL_VM_ACTION enumeration 938
PRL_VM_ACTION_ON_STOP 939
PRL_VM_ACTION_ON_STOP enumeration 939
PRL_VM_ACTION_ON_STOP_PTR 802
PRL_VM_ACTION_ON_STOP_PTR type 802
PRL_VM_ACTION_ON_WINDOW_CLOSE 939
PRL_VM_ACTION_ON_WINDOW_CLOSE enumeration 939
PRL_VM_ACTION_ON_WINDOW_CLOSE_PTR 802
PRL_VM_ACTION_ON_WINDOW_CLOSE_PTR type 802
PRL_VM_ACTION_PTR 802
PRL_VM_ACTION_PTR type 802
PRL_VM_AUTOSTART_OPTION 939
PRL_VM_AUTOSTART_OPTION enumeration 939
PRL_VM_AUTOSTART_OPTION_PTR 802
PRL_VM_AUTOSTART_OPTION_PTR type 802
PRL_VM_AUTOSTOP_OPTION 940
PRL_VM_AUTOSTOP_OPTION enumeration 940
PRL_VM_AUTOSTOP_OPTION_PTR 803
PRL_VM_AUTOSTOP_OPTION_PTR type 803
PRL_VM_BACKUP_FLAGS 940
PRL_VM_BACKUP_FLAGS enumeration 940
PRL_VM_BACKUP_TYPE 941
PRL_VM_BACKUP_TYPE enumeration 941

PRL_VM_BACKUP_TYPE_PTR 803
PRL_VM_BACKUP_TYPE_PTR type 803
PRL_VM_BOOT_ORDER 941
PRL_VM_BOOT_ORDER enumeration 941
PRL_VM_BOOT_ORDER_PTR 803
PRL_VM_BOOT_ORDER_PTR type 803
PRL_VM_COLOR 942
PRL_VM_COLOR enumeration 942
PRL_VM_COLOR_PTR 803
PRL_VM_COLOR_PTR type 803
PRL_VM_COMMIT_FLAGS 942
PRL_VM_COMMIT_FLAGS enumeration 942
PRL_VM_CONFIG_SECTIONS 943
PRL_VM_CONFIG_SECTIONS enumeration 943
PRL_VM_CONFIG_SECTIONS_PTR 803
PRL_VM_CONFIG_SECTIONS_PTR type 803
PRL_VM_CPULIMIT_FULL enumeration member 929
PRL_VM_CPULIMIT_GUEST enumeration member 929
PRL_VM_CPULIMIT_TYPE 943
PRL_VM_CPULIMIT_TYPE enumeration 943
PRL_VM_DEV_EMULATION_TYPE 944
PRL_VM_DEV_EMULATION_TYPE enumeration 944
PRL_VM_DEV_EMULATION_TYPE_PTR 804
PRL_VM_DEV_EMULATION_TYPE_PTR type 804
PRL_VM_EXEC_APP_FILE_DESCRIPTOR 944
PRL_VM_EXEC_APP_FILE_DESCRIPTOR enumeration 944
PRL_VM_EXEC_APP_FILE_DESCRIPTOR_PTR 804
PRL_VM_EXEC_APP_FILE_DESCRIPTOR_PTR type 804
PRL_VM_EXEC_RUN_PROGRAM_MODES 945
PRL_VM_EXEC_RUN_PROGRAM_MODES enumeration 945
PRL_VM_EXEC_RUN_PROGRAM_MODES_PTR 804
PRL_VM_EXEC_RUN_PROGRAM_MODES_PTR type 804
PRL_VM_LOCATION 945
PRL_VM_LOCATION enumeration 945
PRL_VM_LOCATION_PTR 804
PRL_VM_LOCATION_PTR type 804
PRL_VM_MIGRATION_FLAGS 945
PRL_VM_MIGRATION_FLAGS enumeration 945
PRL_VM_MOUSE_CURSOR_STATE 946
PRL_VM_MOUSE_CURSOR_STATE enumeration 946
PRL_VM_MOUSE_CURSOR_STATE_PTR 804
PRL_VM_MOUSE_CURSOR_STATE_PTR type 804
PRL_VM_MOUSE_STATE 947
PRL_VM_MOUSE_STATE enumeration 947
PRL_VM_MOUSE_STATE_PTR 805
PRL_VM_MOUSE_STATE_PTR type 805
PRL_VM_NET_ADAPTER_TYPE 947
PRL_VM_NET_ADAPTER_TYPE enumeration 947
PRL_VM_NET_ADAPTER_TYPE_PTR 805
PRL_VM_NET_ADAPTER_TYPE_PTR type 805
PRL_VM_NET_ADAPTER_WIFI_RULE 947
PRL_VM_NET_ADAPTER_WIFI_RULE enumeration 947
PRL_VM_NET_ADAPTER_WIFI_RULE_PTR 805
PRL_VM_NET_ADAPTER_WIFI_RULE_PTR type 805
PRL_VM_NORMAL_START_FLAGS 947
PRL_VM_NORMAL_START_FLAGS enumeration 947
PRL_VM_NORMAL_START_FLAGS_PTR 805
PRL_VM_NORMAL_START_FLAGS_PTR type 805
PRL_VM_ON_CRASH_ACTION 948
PRL_VM_ON_CRASH_ACTION enumeration 948
PRL_VM_ON_CRASH_ACTION_PTR 805
PRL_VM_ON_CRASH_ACTION_PTR type 805
PRL_VM_ON_CRASH_OPTION 948
PRL_VM_ON_CRASH_OPTION enumeration 948
PRL_VM_ON_CRASH_OPTION_PTR 806
PRL_VM_ON_CRASH_OPTION_PTR type 806
PRL_VM_PRIORITY 948
PRL_VM_PRIORITY enumeration 948
PRL_VM_PRIORITY_PTR 806
PRL_VM_PRIORITY_PTR type 806
PRL_VM_REMOTE_DISPLAY_MODE 949
PRL_VM_REMOTE_DISPLAY_MODE enumeration 949
PRL_VM_REMOTE_DISPLAY_MODE_PTR 806
PRL_VM_REMOTE_DISPLAY_MODE_PTR type 806
PRL_VM_SMARTGUARD_SCHEMA 949
PRL_VM_SMARTGUARD_SCHEMA enumeration 949
PRL_VM_SMARTGUARD_SCHEMA_PTR 806
PRL_VM_SMARTGUARD_SCHEMA_PTR type 806

PRL_VM_START_LOGIN_MODE 949
PRL_VM_START_LOGIN_MODE enumeration 949
PRL_VM_START_LOGIN_MODE_PTR 806
PRL_VM_START_LOGIN_MODE_PTR type 806
PRL_VM_START_MODE 950
PRL_VM_START_MODE enumeration 950
PRL_VM_START_MODE_PTR 807
PRL_VM_START_MODE_PTR type 807
PRL_VM_STATISTICS_FLAGS 950
PRL_VM_STATISTICS_FLAGS enumeration 950
PRL_VM_STATISTICS_FLAGS_PTR 807
PRL_VM_STATISTICS_FLAGS_PTR type 807
PRL_VM_STOP_FLAGS 950
PRL_VM_STOP_FLAGS enumeration 950
PRL_VM_STOP_MODE 951
PRL_VM_STOP_MODE enumeration 951
PRL_VM_STOP_MODE_MASK 1652
PRL_VM_STOP_MODE_MASK macro 1652
PRL_VM_STORE_VALUE_FLAGS 951
PRL_VM_STORE_VALUE_FLAGS enumeration 951
PRL_VM_TOOLS_STATE 952
PRL_VM_TOOLS_STATE enumeration 952
PRL_VM_TOOLS_STATE_PTR 807
PRL_VM_TOOLS_STATE_PTR type 807
PRL_VM_TYPE 952
PRL_VM_TYPE enumeration 952
PRL_VM_TYPE_FLAGS 952
PRL_VM_TYPE_FLAGS enumeration 952
PRL_VM_TYPE_PTR 807
PRL_VM_TYPE_PTR type 807
PRL_VM_WINDOW_MODE 952
PRL_VM_WINDOW_MODE enumeration 952
PRL_VM_WINDOW_MODE_PTR 807
PRL_VM_WINDOW_MODE_PTR type 807
PRL_VMCTG_EVENT 1101
PRL_VMCTG_EVENT enumeration 1101
PRL_VMCTG_EVENT_PTR 834
PRL_VMCTG_EVENT_PTR type 834
PRL_VOID 808
PRL_VOID type 808
PRL_VOID Types 808
PRL_VOID_PTR 808
PRL_VOID_PTR type 808
PRL_VOID_PTR_PTR 808
PRL_VOID_PTR_PTR type 808
PRL_VTD_STATUS_CODE 1101
PRL_VTD_STATUS_CODE enumeration 1101
PRL_VTD_STATUS_CODE_PTR 835
PRL_VTD_STATUS_CODE_PTR type 835
PRL_WARN Macros 1564
PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND 1565
PRL_WARN_BACKUP_DEVICE_IMAGE_NOT_FOUND macro 1565
PRL_WARN_BACKUP_GUEST_SYNCHRONIZATION_FAILED 1565
PRL_WARN_BACKUP_GUEST_SYNCHRONIZATION_FAILED macro 1565
PRL_WARN_BACKUP_GUEST_UNABLE_TO_SYNCHRONIZE 1566
PRL_WARN_BACKUP_GUEST_UNABLE_TO_SYNCHRONIZE macro 1566
PRL_WARN_BACKUP_HAS_NOT_FULL_BACKUP 1566
PRL_WARN_BACKUP_HAS_NOT_FULL_BACKUP macro 1566
PRL_WARN_BACKUP_NON_IMAGE_HDD 1566
PRL_WARN_BACKUP_NON_IMAGE_HDD macro 1566
PRL_WARN_BACKUP_SP_GUEST_SYNCHRONIZATION_FAILED 1566
PRL_WARN_BACKUP_SP_GUEST_SYNCHRONIZATION_FAILED macro 1566
PRL_WARN_CANNOT_MOUNT_BACK_PARTITION 1567
PRL_WARN_CANNOT_MOUNT_BACK_PARTITION macro 1567
PRL_WARN_CANT_CONNECT_VOLUME 1567
PRL_WARN_CANT_CONNECT_VOLUME macro 1567
PRL_WARN_COMPACT_DESTRUCTIVE 1567
PRL_WARN_COMPACT_DESTRUCTIVE macro 1567
PRL_WARN_COMPACT_THROUGH_COPY 1567

PRL_WARN_COMPACT_THROUGH_COPY macro 1567

PRL_WARN_DELETE_NO_CONFIG_UNREGISTER 1567

PRL_WARN_DELETE_NO_CONFIG_UNREGISTER macro 1567

PRL_WARN_DEV_USB_CONNECT_FAILED 1568

PRL_WARN_DEV_USB_CONNECT_FAILED macro 1568

PRL_WARN_DEV_USB_OPEN_FAILED 1568

PRL_WARN_DEV_USB_OPEN_FAILED macro 1568

PRL_WARN_FAILED_TO_START_VNC_SERVER 1568

PRL_WARN_FAILED_TO_START_VNC_SERVER macro 1568

PRL_WARN_FAT32_FILE_OVERRUN 1568

PRL_WARN_FAT32_FILE_OVERRUN macro 1568

PRL_WARN_GOING_TO_TAKE_SMART_GUARD_SNAPSHOT 1569

PRL_WARN_GOING_TO_TAKE_SMART_GUARD_SNAPSHOT macro 1569

PRL_WARN_HDD_ERROR 1569

PRL_WARN_HDD_ERROR macro 1569

PRL_WARN_IMAGE_IS_STOLEN 1569

PRL_WARN_IMAGE_IS_STOLEN macro 1569

PRL_WARN_LAN_NOT_CONNECTED 1569

PRL_WARN_LAN_NOT_CONNECTED macro 1569

PRL_WARN_LICENSE_RESTRICTED_RKU 1570

PRL_WARN_LICENSE_RESTRICTED_RKU macro 1570

PRL_WARN_LINK_TO_OUTSIDED_HDD_WILL_BE_DROPPED_AFTER_CLONE 1570

PRL_WARN_LINK_TO_OUTSIDED_HDD_WILL_BE_DROPPED_AFTER_CLONE macro 1570

PRL_WARN_NESTED_VIRT_NOT_ENABLED 1570

PRL_WARN_NESTED_VIRT_NOT_ENABLED macro 1570

PRL_WARN_NO_SHARED_NETWORK_FOR_OFFLINE_MANAGEMENT 1570

PRL_WARN_NO_SHARED_NETWORK_FOR_OFFLINE_MANAGEMENT macro 1570

PRL_WARN_NOT_SYMMETRIC_CPU 1571

PRL_WARN_NOT_SYMMETRIC_CPU macro 1571

PRL_WARN_OUTSIDED_HDD_WILL_BE_ONLY_LINKED_AFTER_CLONE 1571

PRL_WARN_OUTSIDED_HDD_WILL_BE_ONLY_LINKED_AFTER_CLONE macro 1571

PRL_WARN_PREPARE_FOR_HIBERNATE_UNABLE_SUSPEND_DO_STOP 1571

PRL_WARN_PREPARE_FOR_HIBERNATE_UNABLE_SUSPEND_DO_STOP macro 1571

PRL_WARN_TARGET_HOST_DISK_IS_FULL 1571

PRL_WARN_TARGET_HOST_DISK_IS_FULL macro 1571

PRL_WARN_TOO_LOW_HDD_SIZE 1572

PRL_WARN_TOO_LOW_HDD_SIZE macro 1572

PRL_WARN_UNABLE_OPEN_DEVICE 1572

PRL_WARN_UNABLE_OPEN_DEVICE macro 1572

PRL_WARN_UNABLE_OPEN_DEVICE_NAME 1572

PRL_WARN_UNABLE_OPEN_DEVICE_NAME macro 1572

PRL_WARN_UNABLE_OPEN_DEVICE_ON_START_VM 1572

PRL_WARN_UNABLE_OPEN_DEVICE_ON_START_VM macro 1572

PRL_WARN_UNKNOWN_HDD_REQUEST_TYPE 1572

PRL_WARN_UNKNOWN_HDD_REQUEST_TYPE macro 1572

PRL_WARN_VERBOSE_LOGGING 1573

PRL_WARN_VERBOSE_LOGGING macro 1573

PRL_WARN_VM_BOOTCAMP_MODE 1573

PRL_WARN_VM_BOOTCAMP_MODE macro 1573

PRL_WARN_VM_DISCONNECT_SATA_HDD 1573

PRL_WARN_VM_DISCONNECT_SATA_HDD macro 1573

PRL_WARN_VM_DVD_DISCONNECT_LOCKED_DEV 1573

PRL_WARN_VM_DVD_DISCONNECT_LOCKED_DEV macro 1573

PRL_WARN_VM_DVD_DISCONNECTED_BY_GUEST 1574

PRL_WARN_VM_DVD_DISCONNECTED_BY_GUEST macro 1574

PRL_WARN_VM_PD3_COMPAT_MODE 1574

PRL_WARN_VM_PD3_COMPAT_MODE macro 1574

PRL_WARN_VM_PD3_COMPAT_NO_MOUSE 1574

PRL_WARN_VM_PD3_COMPAT_NO_MOUSE macro 1574

PRL_WARN_VM_SYNC_DEFAULT_PRINTER_DISABLED 1574

PRL_WARN_VM_SYNC_DEFAULT_PRINTER_DISABLED macro 1574

PRL_WARN_WINDOWS_UPDATE_WORKING 1574

PRL_WARN_WINDOWS_UPDATE_WORKING macro 1574
PRL_WIN_EXTENDED_PARTITION 1653
PRL_WIN_EXTENDED_PARTITION macro 1653
PRL_WINDOW_CORNERS 1102
PRL_WINDOW_CORNERS enumeration 1102
PRL_WINDOW_CORNERS_PTR 835
PRL_WINDOW_CORNERS_PTR type 835
PRL_WINDOW_EDGE_MASK 1102
PRL_WINDOW_EDGE_MASK enumeration 1102
PRL_WINDOW_EDGE_MASK_PTR 835
PRL_WINDOW_EDGE_MASK_PTR type 835
PRL_WINDOWS_PARTITION_FAT 1653
PRL_WINDOWS_PARTITION_FAT macro 1653
PRL_WINDOWS_PARTITION_FAT32 1653
PRL_WINDOWS_PARTITION_FAT32 macro 1653
PRL_WINDOWS_PARTITION_FAT32_LBA 1653
PRL_WINDOWS_PARTITION_FAT32_LBA macro 1653
PRL_WINDOWS_PARTITION_NTFS 1653
PRL_WINDOWS_PARTITION_NTFS macro 1653
PRL_WNG_NO_OPERATION_SYSTEM_INSTALLED 1654
PRL_WNG_NO_OPERATION_SYSTEM_INSTALLED macro 1654
PRL_WNG_START_VM_ON_MOUNTED_DISK 1654
PRL_WNG_START_VM_ON_MOUNTED_DISK macro 1654
PRL_WRN Macros 1575
PRL_WRN_IT_CANT_RESIZE_LDM_DISK 1575
PRL_WRN_IT_CANT_RESIZE_LDM_DISK macro 1575
PRL_WRN_IT_CANT_RESIZE_VOLUME 1575
PRL_WRN_IT_CANT_RESIZE_VOLUME macro 1575
PRL_WRN_IT_EMPTY_MBR 1575
PRL_WRN_IT_EMPTY_MBR macro 1575
PRL_WRN_IT_FIRST_WARNING 1576
PRL_WRN_IT_FIRST_WARNING macro 1576
PRL_WRN_IT_VOLUME_RESIZING 1576
PRL_WRN_IT_VOLUME_RESIZING macro 1576
PrlAcl_GetAccessForOthers 10
PrlAcl_GetAccessForOthers function 10
PrlAcl_GetOwnerName 10
PrlAcl_GetOwnerName function 10
PrlAcl_IsAllowed 11
PrlAcl_IsAllowed function 11
PrlAcl_IsCurrentSessionOwner 12
PrlAcl_IsCurrentSessionOwner function 12
PrlAcl_SetAccessForOthers 12
PrlAcl_SetAccessForOthers function 12
PrlApi_AddRemoteDevCommandBuffer@PRL_HANDLE@PRL_VOID_PTR@PRL_UINT32
728
PrlApi_AddRemoteDevCommandParam@PRL_HANDLE@PRL_UINT32
728
PrlApi_CreateHandlesList 728
PrlApi_CreateHandlesList function 728
PrlApi_CreateOpTypeList 729
PrlApi_CreateOpTypeList function 729
PrlApi_CreateProblemReport 729
PrlApi_CreateProblemReport function 729
PrlApi_CreateStringsList 730
PrlApi_CreateStringsList function 730
PrlApi_Deinit 731
PrlApi_Deinit function 731
PrlApi_GetAppMode 731
PrlApi_GetAppMode function 731
PrlApi_GetCrashDumpsPath 732
PrlApi_GetCrashDumpsPath function 732
PrlApi_GetDefaultOsVersion 732
PrlApi_GetDefaultOsVersion function 732
PrlApi_GetMessageType 733
PrlApi_GetMessageType function 733
PrlApi_GetRecommendMinVmMem 733
PrlApi_GetRecommendMinVmMem function 733
PrlApi_GetResultDescription 734
PrlApi_GetResultDescription function 734
PrlApi_GetSupportedOsesTypes 735
PrlApi_GetSupportedOsesTypes function 735
PrlApi_GetSupportedOsesVersions 735
PrlApi_GetSupportedOsesVersions function 735
PrlApi_GetVersion 736
PrlApi_GetVersion function 736
PrlApi_GuestToString 736
PrlApi_GuestToString function 736

PrlApi_Init 737
PrlApi_Init function 737
PrlApi_InitCrashHandler 739
PrlApi_InitCrashHandler function 739
PrlApi_InitEx 739
PrlApi_InitEx function 739
PrlApi_MsgCanBeIgnored 740
PrlApi_MsgCanBeIgnored function 740
PrlApi_SendPackedProblemReport 741
PrlApi_SendPackedProblemReport function 741
PrlApi_SendProblemReport 742
PrlApi_SendProblemReport function 742
PrlApi_SwitchConsoleLogging 743
PrlApi_SwitchConsoleLogging function 743
PrlAppliance_Create 743
PrlAppliance_Create function 743
PrlBackupResult_GetBackupUuid 13
PrlBackupResult_GetBackupUuid function 13
PrlBootDev_GetIndex 15
PrlBootDev_GetIndex function 15
PrlBootDev_GetSequenceIndex 15
PrlBootDev_GetSequenceIndex function 15
PrlBootDev_GetType 16
PrlBootDev_GetType function 16
PrlBootDev_IsInUse 17
PrlBootDev_IsInUse function 17
PrlBootDev_Remove 17
PrlBootDev_Remove function 17
PrlBootDev_SetIndex 18
PrlBootDev_SetIndex function 18
PrlBootDev_SetInUse 19
PrlBootDev_SetInUse function 19
PrlBootDev_SetSequenceIndex 19
PrlBootDev_SetSequenceIndex function 19
PrlBootDev_SetType 21
PrlBootDev_SetType function 21
PrlBuffer 775
PrlBuffer_Free 776
PrlBuffer_Free function 776
PrlCpuFeatures_Create 22
PrlCpuFeatures_Create function 22
PrlCpuFeatures_GetValue 23
PrlCpuFeatures_GetValue function 23
PrlCpuFeatures_SetValue 23
PrlCpuFeatures_SetValue function 23
PrlCPUPool_GetCpuFeaturesMask 24
PrlCPUPool_GetCpuFeaturesMask function 24
PrlCPUPool_GetName 25
PrlCPUPool_GetName function 25
PrlCPUPool_GetVendor 25
PrlCPUPool_GetVendor function 25
PrlCt_Reinstall 445
PrlCt_Reinstall function 445
PrlCtTemplate_GetCpuMode 26
PrlCtTemplate_GetCpuMode function 26
PrlCtTemplate_GetDescription 27
PrlCtTemplate_GetDescription function 27
PrlCtTemplate_GetName 28
PrlCtTemplate_GetName function 28
PrlCtTemplate_GetOsTemplate 28
PrlCtTemplate_GetOsTemplate function 28
PrlCtTemplate_GetOsType 29
PrlCtTemplate_GetOsType function 29
PrlCtTemplate_GetOsVersion 30
PrlCtTemplate_GetOsVersion function 30
PrlCtTemplate_GetType 30
PrlCtTemplate_GetType function 30
PrlCtTemplate_GetVersion 31
PrlCtTemplate_GetVersion function 31
PrlCtTemplate_IsCached 31
PrlCtTemplate_IsCached function 31
PrlDbg_EventTypeToString 745
PrlDbg_EventTypeToString function 745
PrlDbg_GetHandlesNum 746
PrlDbg_GetHandlesNum function 746
PrlDbg_HandleTypeToString 746
PrlDbg_HandleTypeToString function 746
PrlDbg_PrlResultToString 747

PrlDbg_PrlResultToString function 747
PrlDisk_CreateDisk_Local 433
PrlDisk_CreateDisk_Local function 433
PrlDisk_Free 434
PrlDisk_Free function 434
PrlDisk_GetDiskInfo 434
PrlDisk_GetDiskInfo function 434
PrlDisk_OpenDisk_Local 435
PrlDisk_OpenDisk_Local function 435
PrlDisk_Read 436
PrlDisk_Read function 436
PrlDisk_SwitchToState 436
PrlDisk_SwitchToState function 436
PrlDisk_WaitForCompletion 437
PrlDisk_WaitForCompletion function 437
PrlDisk_Write 438
PrlDisk_Write function 438
PrlDiskMap_GetGranularity 438
PrlDiskMap_GetGranularity function 438
PrlDiskMap_GetSize 439
PrlDiskMap_GetSize function 439
PrlDiskMap_Read 439
PrlDiskMap_Read function 439
PrlDiskOpenPolicy_CreateOffset 440
PrlDiskOpenPolicy_CreateOffset function 440
PrlDispCfg_ArePluginsEnabled 35
PrlDispCfg_ArePluginsEnabled function 35
PrlDispCfg_CanChangeDefaultSettings 36
PrlDispCfg_CanChangeDefaultSettings function 36
PrlDispCfg_EnablePlugins 36
PrlDispCfg_EnablePlugins function 36
PrlDispCfg_GetBackupTimeout 37
PrlDispCfg_GetBackupTimeout function 37
PrlDispCfg_GetBackupTmpDir 37
PrlDispCfg_GetBackupTmpDir function 37
PrlDispCfg_GetBackupUserLogin 38
PrlDispCfg_GetBackupUserLogin function 38
PrlDispCfg_GetConfirmationsList 39
PrlDispCfg_GetConfirmationsList function 39
PrlDispCfg_GetCpuFeaturesMaskEx 39
PrlDispCfg_GetCpuFeaturesMaskEx function 39
PrlDispCfg_GetCpuPool 40
PrlDispCfg_GetCpuPool function 40
PrlDispCfg_GetDefaultBackupDirectory 40
PrlDispCfg_GetDefaultBackupDirectory function 40
PrlDispCfg_GetDefaultBackupServer 41
PrlDispCfg_GetDefaultBackupServer function 41
PrlDispCfg_GetDefaultCtDir 42
PrlDispCfg_GetDefaultCtDir function 42
PrlDispCfg_GetDefaultVmDir 42
PrlDispCfg_GetDefaultVmDir function 42
PrlDispCfg_GetDefaultVNCHostName 43
PrlDispCfg_GetDefaultVNCHostName function 43
PrlDispCfg_GetMaxReservMemLimit 44
PrlDispCfg_GetMaxReservMemLimit function 44
PrlDispCfg_GetMaxVmMem 45
PrlDispCfg_GetMaxVmMem function 45
PrlDispCfg_GetMinReservMemLimit 45
PrlDispCfg_GetMinReservMemLimit function 45
PrlDispCfg_GetMinSecurityLevel 46
PrlDispCfg_GetMinSecurityLevel function 46
PrlDispCfg_GetMinVmMem 47
PrlDispCfg_GetMinVmMem function 47
PrlDispCfg_GetRecommendMaxVmMem 48
PrlDispCfg_GetRecommendMaxVmMem function 48
PrlDispCfg_GetReservedMemLimit 49
PrlDispCfg_GetReservedMemLimit function 49
PrlDispCfg_GetUsbIdentity 49
PrlDispCfg_GetUsbIdentity function 49
PrlDispCfg_GetUsbIdentityCount 50
PrlDispCfg_GetUsbIdentityCount function 50
PrlDispCfg_GetVmCpuLimitType 51
PrlDispCfg_GetVmCpuLimitType function 51
PrlDispCfg_GetVNCBasePort 51
PrlDispCfg_GetVNCBasePort function 51
PrlDispCfg_IsAdjustMemAuto 52
PrlDispCfg_IsAdjustMemAuto function 52
PrlDispCfg_IsAllowMultiplePMC 53

PrlDispCfg_IsAllowMultiplePMC function 53
PrlDispCfg_IsBackupUserPasswordEnabled 53
PrlDispCfg_IsBackupUserPasswordEnabled function 53
PrlDispCfg_IsLogRotationEnabled 54
PrlDispCfg_IsLogRotationEnabled function 54
PrlDispCfg_IsSendStatisticReport 54
PrlDispCfg_IsSendStatisticReport function 54
PrlDispCfg_IsVerboseLogEnabled 55
PrlDispCfg_IsVerboseLogEnabled function 55
PrlDispCfg_SetAdjustMemAuto 55
PrlDispCfg_SetAdjustMemAuto function 55
PrlDispCfg_SetAllowMultiplePMC 56
PrlDispCfg_SetAllowMultiplePMC function 56
PrlDispCfg_SetBackupTimeout 57
PrlDispCfg_SetBackupTimeout function 57
PrlDispCfg_SetBackupTmpDir 57
PrlDispCfg_SetBackupTmpDir function 57
PrlDispCfg_SetBackupUserLogin 58
PrlDispCfg_SetBackupUserLogin function 58
PrlDispCfg_SetBackupUserPassword 58
PrlDispCfg_SetBackupUserPassword function 58
PrlDispCfg_SetBackupUserPasswordEnabled 59
PrlDispCfg_SetBackupUserPasswordEnabled function 59
PrlDispCfg_SetCanChangeDefaultSettings 59
PrlDispCfg_SetCanChangeDefaultSettings function 59
PrlDispCfg_SetConfirmationsList 60
PrlDispCfg_SetConfirmationsList function 60
PrlDispCfg_SetCpuFeaturesMaskEx 61
PrlDispCfg_SetCpuFeaturesMaskEx function 61
PrlDispCfg_SetDefaultBackupDirectory 61
PrlDispCfg_SetDefaultBackupDirectory function 61
PrlDispCfg_SetDefaultBackupServer 62
PrlDispCfg_SetDefaultBackupServer function 62
PrlDispCfg_SetDefaultVNCHostName 62
PrlDispCfg_SetDefaultVNCHostName function 62
PrlDispCfg_SetLogRotationEnabled 63
PrlDispCfg_SetLogRotationEnabled function 63
PrlDispCfg_SetMaxReservMemLimit 64
PrlDispCfg_SetMaxReservMemLimit function 64
PrlDispCfg_SetMaxVmMem 64
PrlDispCfg_SetMaxVmMem function 64
PrlDispCfg_SetMinReservMemLimit 65
PrlDispCfg_SetMinReservMemLimit function 65
PrlDispCfg_SetMinSecurityLevel 66
PrlDispCfg_SetMinSecurityLevel function 66
PrlDispCfg_SetMinVmMem 67
PrlDispCfg_SetMinVmMem function 67
PrlDispCfg_SetRecommendMaxVmMem 68
PrlDispCfg_SetRecommendMaxVmMem function 68
PrlDispCfg_SetReservedMemLimit 68
PrlDispCfg_SetReservedMemLimit function 68
PrlDispCfg_SetSendStatisticReport 69
PrlDispCfg_SetSendStatisticReport function 69
PrlDispCfg_SetUsbIdentAssociation 70
PrlDispCfg_SetUsbIdentAssociation function 70
PrlDispCfg_SetVerboseLogEnabled 71
PrlDispCfg_SetVerboseLogEnabled function 71
PrlDispCfg_SetVmCpuLimitType 71
PrlDispCfg_SetVmCpuLimitType function 71
PrlDispCfg_SetVNCBasePort 72
PrlDispCfg_SetVNCBasePort function 72
PrlEvent_CanBeIgnored 73
PrlEvent_CanBeIgnored function 73
PrlEvent_CreateAnswerEvent 74
PrlEvent_CreateAnswerEvent function 74
PrlEvent_GetDataPtr 80
PrlEvent_GetDataPtr function 80
PrlEvent_GetErrCode 81
PrlEvent_GetErrCode function 81
PrlEvent_GetErrString 81
PrlEvent_GetErrString function 81
PrlEvent_GetIssuerId 82
PrlEvent_GetIssuerId function 82
PrlEvent_GetIssuerType 83
PrlEvent_GetIssuerType function 83
PrlEvent_GetJob 83
PrlEvent_GetJob function 83
PrlEvent_GetParam 84

PrlEvent_GetParam function 84
PrlEvent_GetParamByName 85
PrlEvent_GetParamByName function 85
PrlEvent_GetParamsCount 85
PrlEvent_GetParamsCount function 85
PrlEvent_GetServer 86
PrlEvent_GetServer function 86
PrlEvent_GetType 86
PrlEvent_GetType function 86
PrlEvent_GetVm 87
PrlEvent_GetVm function 87
PrlEvent_IsAnswerRequired 88
PrlEvent_IsAnswerRequired function 88
PrlEvtPrm_GetBuffer 89
PrlEvtPrm_GetBuffer function 89
PrlEvtPrm_GetName 90
PrlEvtPrm_GetName function 90
PrlEvtPrm_GetType 90
PrlEvtPrm_GetType function 90
PrlEvtPrm_ToBoolean 91
PrlEvtPrm_ToBoolean function 91
PrlEvtPrm_ToCData 91
PrlEvtPrm_ToCData function 91
PrlEvtPrm_ToHandle 92
PrlEvtPrm_ToHandle function 92
PrlEvtPrm_ToInt32 93
PrlEvtPrm_ToInt32 function 93
PrlEvtPrm_ToInt64 93
PrlEvtPrm_ToInt64 function 93
PrlEvtPrm_ToString 94
PrlEvtPrm_ToString function 94
PrlEvtPrm_ToUint32 94
PrlEvtPrm_ToUint32 function 94
PrlEvtPrm_ToUint64 95
PrlEvtPrm_ToUint64 function 95
PrlFirewallRule_Create 96
PrlFirewallRule_Create function 96
PrlFirewallRule_GetLocalNetAddress 96
PrlFirewallRule_GetLocalNetAddress function 96
PrlFirewallRule_GetLocalPort 97
PrlFirewallRule_GetLocalPort function 97
PrlFirewallRule_GetProtocol 98
PrlFirewallRule_GetProtocol function 98
PrlFirewallRule_GetRemoteNetAddress 98
PrlFirewallRule_GetRemoteNetAddress function 98
PrlFirewallRule_GetRemotePort 99
PrlFirewallRule_GetRemotePort function 99
PrlFirewallRule_SetLocalNetAddress 99
PrlFirewallRule_SetLocalNetAddress function 99
PrlFirewallRule_SetLocalPort 100
PrlFirewallRule_SetLocalPort function 100
PrlFirewallRule_SetProtocol 100
PrlFirewallRule_SetProtocol function 100
PrlFirewallRule_SetRemoteNetAddress 101
PrlFirewallRule_SetRemoteNetAddress function 101
PrlFirewallRule_SetRemotePort 101
PrlFirewallRule_SetRemotePort function 101
PrlFoundVmInfo_GetConfigPath 102
PrlFoundVmInfo_GetConfigPath function 102
PrlFoundVmInfo_GetName 103
PrlFoundVmInfo_GetName function 103
PrlFoundVmInfo_GetOSVersion 104
PrlFoundVmInfo_GetOSVersion function 104
PrlFoundVmInfo_IsOldConfig 104
PrlFoundVmInfo_IsOldConfig function 104
PrlFoundVmInfo_IsTemplate 105
PrlFoundVmInfo_IsTemplate function 105
PrlFsEntry_GetAbsolutePath 203
PrlFsEntry_GetAbsolutePath function 203
PrlFsEntry_GetLastModifiedDate 203
PrlFsEntry_GetLastModifiedDate function 203
PrlFsEntry_GetPermissions 204
PrlFsEntry_GetPermissions function 204
PrlFsEntry_GetRelativeName 205
PrlFsEntry_GetRelativeName function 205
PrlFsEntry_GetSize 205
PrlFsEntry_GetSize function 205
PrlFsEntry_GetType 206

PrlFsEntry_GetType function 206
PrlFsInfo_GetChildEntriesCount 207
PrlFsInfo_GetChildEntriesCount function 207
PrlFsInfo_GetChildEntry 207
PrlFsInfo_GetChildEntry function 207
PrlFsInfo_GetFsType 208
PrlFsInfo_GetFsType function 208
PrlFsInfo_GetParentEntry 209
PrlFsInfo_GetParentEntry function 209
PrlFsInfo_GetType 209
PrlFsInfo_GetType function 209
PrlHandle_AddRef 748
PrlHandle_AddRef function 748
PrlHandle_Free 748
PrlHandle_Free function 748
PrlHandle_GetChannelStatistics 749
PrlHandle_GetChannelStatistics function 749
PrlHandle_GetPackageId 750
PrlHandle_GetPackageId function 750
PrlHandle_GetType 750
PrlHandle_GetType function 750
PrlHandle_RegEventHandler 751
PrlHandle_RegEventHandler function 751
PrlHandle_UnregEventHandler 751
PrlHandle_UnregEventHandler function 751
PrlHndlList_AddItem 109
PrlHndlList_AddItem function 109
PrlHndlList_GetItem 109
PrlHndlList_GetItem function 109
PrlHndlList_GetItemsCount 110
PrlHndlList_GetItemsCount function 110
PrlHndlList_RemoveItem 110
PrlHndlList_RemoveItem function 110
PrlIPPrivNet_Create 134
PrlIPPrivNet_Create function 134
PrlIPPrivNet_GetName 134
PrlIPPrivNet_GetName function 134
PrlIPPrivNet_GetNetAddresses 135
PrlIPPrivNet_GetNetAddresses function 135
PrlIPPrivNet_IsGlobal 136
PrlIPPrivNet_IsGlobal function 136
PrlIPPrivNet_SetGlobal 136
PrlIPPrivNet_SetGlobal function 136
PrlIPPrivNet_SetName 137
PrlIPPrivNet_SetName function 137
PrlIPPrivNet_SetNetAddresses 137
PrlIPPrivNet_SetNetAddresses function 137
PrlJob_Cancel 139
PrlJob_Cancel function 139
PrlJob_GetDataPtr 139
PrlJob_GetDataPtr function 139
PrlJob_GetError 140
PrlJob_GetError function 140
PrlJob_GetOpCode 141
PrlJob_GetOpCode function 141
PrlJob_GetProgress 141
PrlJob_GetProgress function 141
PrlJob_GetResult 142
PrlJob_GetResult function 142
PrlJob_GetRetCode 143
PrlJob_GetRetCode function 143
PrlJob_GetStatus 143
PrlJob_GetStatus function 143
PrlJob_IsRequestWasSent 144
PrlJob_IsRequestWasSent function 144
PrlJob_Wait 144
PrlJob_Wait function 144
PrlLic_GetCompanyName 146
PrlLic_GetCompanyName function 146
PrlLic_GetLicenseKey 147
PrlLic_GetLicenseKey function 147
PrlLic_GetStatus 147
PrlLic_GetStatus function 147
PrlLic_GetUserName 148
PrlLic_GetUserName function 148
PrlLic_IsValid 149
PrlLic_IsValid function 149
PrlLoginResponse_GetHostOsVersion 150

PrlLoginResponse_GetHostOsVersion function 150
PrlLoginResponse_GetProductVersion 150
PrlLoginResponse_GetProductVersion function 150
PrlLoginResponse_GetRunningTaskByIndex 151
PrlLoginResponse_GetRunningTaskByIndex function 151
PrlLoginResponse_GetRunningTaskCount 152
PrlLoginResponse_GetRunningTaskCount function 152
PrlLoginResponse_GetServerUuid 152
PrlLoginResponse_GetServerUuid function 152
PrlLoginResponse_GetSessionUuid 153
PrlLoginResponse_GetSessionUuid function 153
PrlNetSvc_GetStatus 154
PrlNetSvc_GetStatus function 154
PrlNetworkClass_Create 155
PrlNetworkClass_Create function 155
PrlNetworkClass_GetClassId 155
PrlNetworkClass_GetClassId function 155
PrlNetworkClass_GetNetworkList 156
PrlNetworkClass_GetNetworkList function 156
PrlNetworkClass_SetClassId 156
PrlNetworkClass_SetClassId function 156
PrlNetworkClass_SetNetworkList 157
PrlNetworkClass_SetNetworkList function 157
PrlNetworkRate_Create 158
PrlNetworkRate_Create function 158
PrlNetworkRate_GetClassId 158
PrlNetworkRate_GetClassId function 158
PrlNetworkRate_GetRate 159
PrlNetworkRate_GetRate function 159
PrlNetworkShapingBandwidthEntry_Create 166
PrlNetworkShapingBandwidthEntry_Create function 166
PrlNetworkShapingBandwidthEntry_GetBandwidth 167
PrlNetworkShapingBandwidthEntry_GetBandwidth function 167
PrlNetworkShapingBandwidthEntry_GetDevice 168
PrlNetworkShapingBandwidthEntry_GetDevice function 168
PrlNetworkShapingBandwidthEntry_SetBandwidth 168
PrlNetworkShapingBandwidthEntry_SetBandwidth function 168
PrlNetworkShapingBandwidthEntry_SetDevice 169
PrlNetworkShapingBandwidthEntry_SetDevice function 169
PrlNetworkShapingConfig_GetNetworkDeviceBandwidthList 170
PrlNetworkShapingConfig_GetNetworkDeviceBandwidthList function 170
PrlNetworkShapingConfig_GetNetworkShapingList 170
PrlNetworkShapingConfig_GetNetworkShapingList function 170
PrlNetworkShapingConfig_IsEnabled 171
PrlNetworkShapingConfig_IsEnabled function 171
PrlNetworkShapingConfig_SetEnabled 171
PrlNetworkShapingConfig_SetEnabled function 171
PrlNetworkShapingConfig_SetNetworkDeviceBandwidthList 172
PrlNetworkShapingConfig_SetNetworkDeviceBandwidthList function 172
PrlNetworkShapingConfig_SetNetworkShapingList 172
PrlNetworkShapingConfig_SetNetworkShapingList function 172
PrlNetworkShapingEntry_Create 160
PrlNetworkShapingEntry_Create function 160
PrlNetworkShapingEntry_GetClassId 161
PrlNetworkShapingEntry_GetClassId function 161
PrlNetworkShapingEntry_GetDevice 161
PrlNetworkShapingEntry_GetDevice function 161
PrlNetworkShapingEntry_GetRate 162
PrlNetworkShapingEntry_GetRate function 162
PrlNetworkShapingEntry_GetRateMPU 162
PrlNetworkShapingEntry_GetRateMPU function 162
PrlNetworkShapingEntry_GetTotalRate 163
PrlNetworkShapingEntry_GetTotalRate function 163
PrlNetworkShapingEntry_SetClassId 163
PrlNetworkShapingEntry_SetClassId function 163
PrlNetworkShapingEntry_SetDevice 164
PrlNetworkShapingEntry_SetDevice function 164
PrlNetworkShapingEntry_SetRate 164
PrlNetworkShapingEntry_SetRate function 164
PrlNetworkShapingEntry_SetRateMPU 165
PrlNetworkShapingEntry_SetRateMPU function 165
PrlNetworkShapingEntry_SetTotalRate 165
PrlNetworkShapingEntry_SetTotalRate function 165
PrlOffmgmtService_Create 173
PrlOffmgmtService_Create function 173
PrlOffmgmtService_GetName 174
PrlOffmgmtService_GetName function 174

PrlOffmgmtService_GetPort 175
PrlOffmgmtService_GetPort function 175
PrlOffmgmtService_IsUsedByDefault 175
PrlOffmgmtService_IsUsedByDefault function 175
PrlOffmgmtService_SetName 176
PrlOffmgmtService_SetName function 176
PrlOffmgmtService_SetPort 176
PrlOffmgmtService_SetPort function 176
PrlOffmgmtService_SetUsedByDefault 177
PrlOffmgmtService_SetUsedByDefault function 177
PrlOpTypeList_AddItem 178
PrlOpTypeList_AddItem function 178
PrlOpTypeList_GetItem 178
PrlOpTypeList_GetItem function 178
PrlOpTypeList_GetItemsCount 179
PrlOpTypeList_GetItemsCount function 179
PrlOpTypeList_GetTypeSize 179
PrlOpTypeList_GetTypeSize function 179
PrlOpTypeList_RemoveItem 180
PrlOpTypeList_RemoveItem function 180
PrlOsesMatrix_GetDefaultOsVersion 106
PrlOsesMatrix_GetDefaultOsVersion function 106
PrlOsesMatrix_GetSupportedOsesTypes 106
PrlOsesMatrix_GetSupportedOsesTypes function 106
PrlOsesMatrix_GetSupportedOsesVersions 107
PrlOsesMatrix_GetSupportedOsesVersions function 107
PrlOsesMatrix_GetSupportType 108
PrlOsesMatrix_GetSupportType function 108
PrlPluginInfo_GetCopyright 181
PrlPluginInfo_GetCopyright function 181
PrlPluginInfo_GetId 181
PrlPluginInfo_GetId function 181
PrlPluginInfo_GetLongDescription 182
PrlPluginInfo_GetLongDescription function 182
PrlPluginInfo_GetShortDescription 183
PrlPluginInfo_GetShortDescription function 183
PrlPluginInfo_GetVendor 183
PrlPluginInfo_GetVendor function 183
PrlPluginInfo_GetVersion 184
PrlPluginInfo_GetVersion function 184
PrlPortFwd_Create 186
PrlPortFwd_Create function 186
PrlPortFwd_GetIncomingPort 186
PrlPortFwd_GetIncomingPort function 186
PrlPortFwd_GetRedirectIPAddress 187
PrlPortFwd_GetRedirectIPAddress function 187
PrlPortFwd_GetRedirectPort 187
PrlPortFwd_GetRedirectPort function 187
PrlPortFwd_GetRedirectVm 188
PrlPortFwd_GetRedirectVm function 188
PrlPortFwd_GetRuleName 189
PrlPortFwd_GetRuleName function 189
PrlPortFwd_SetIncomingPort 189
PrlPortFwd_SetIncomingPort function 189
PrlPortFwd_SetRedirectIPAddress 190
PrlPortFwd_SetRedirectIPAddress function 190
PrlPortFwd_SetRedirectPort 190
PrlPortFwd_SetRedirectPort function 190
PrlPortFwd_SetRedirectVm 191
PrlPortFwd_SetRedirectVm function 191
PrlPortFwd_SetRuleName 191
PrlPortFwd_SetRuleName function 191
PrlReport_Assembly 192
PrlReport_Assembly function 192
PrlReport_AsString 193
PrlReport_AsString function 193
PrlReport_GetArchiveFileName 194
PrlReport_GetArchiveFileName function 194
PrlReport_GetData 194
PrlReport_GetData function 194
PrlReport_GetDescription 195
PrlReport_GetDescription function 195
PrlReport_GetReason 196
PrlReport_GetReason function 196
PrlReport_GetScheme 196
PrlReport_GetScheme function 196
PrlReport_GetType 197
PrlReport_GetType function 197

PrlReport_GetUserEmail 197
PrlReport_GetUserEmail function 197
PrlReport_GetUserName 198
PrlReport_GetUserName function 198
PrlReport_Send 198
PrlReport_Send function 198
PrlReport_SetDescription 200
PrlReport_SetDescription function 200
PrlReport_SetReason 200
PrlReport_SetReason function 200
PrlReport_SetType 201
PrlReport_SetType function 201
PrlReport_SetUserEmail 201
PrlReport_SetUserEmail function 201
PrlReport_SetUserName 202
PrlReport_SetUserName function 202
PrlResult_GetParam 210
PrlResult_GetParam function 210
PrlResult_GetParamAsString 213
PrlResult_GetParamAsString function 213
PrlResult_GetParamByIndex 213
PrlResult_GetParamByIndex function 213
PrlResult_GetParamByIndexAsString 215
PrlResult_GetParamByIndexAsString function 215
PrlResult_GetParamsCount 216
PrlResult_GetParamsCount function 216
PrlRunningTask_GetTaskParametersAsString 217
PrlRunningTask_GetTaskParametersAsString function 217
PrlRunningTask_GetTaskType 218
PrlRunningTask_GetTaskType function 218
PrlRunningTask_GetTaskUuid 218
PrlRunningTask_GetTaskUuid function 218
PrlShare_GetDescription 351
PrlShare_GetDescription function 351
PrlShare_GetName 351
PrlShare_GetName function 351
PrlShare_GetPath 352
PrlShare_GetPath function 352
PrlShare_IsEnabled 353
PrlShare_IsEnabled function 353
PrlShare_IsReadOnly 353
PrlShare_IsReadOnly function 353
PrlShare_Remove 354
PrlShare_Remove function 354
PrlShare_SetDescription 354
PrlShare_SetDescription function 354
PrlShare_SetEnabled 355
PrlShare_SetEnabled function 355
PrlShare_SetName 356
PrlShare_SetName function 356
PrlShare_SetPath 356
PrlShare_SetPath function 356
PrlShare_SetReadOnly 357
PrlShare_SetReadOnly function 357
PrlSrv_AddIPPrivateNetwork 223
PrlSrv_AddIPPrivateNetwork function 223
PrlSrv_AddVirtualNetwork 224
PrlSrv_AddVirtualNetwork function 224
PrlSrv_AttachToLostTask 224
PrlSrv_AttachToLostTask function 224
PrlSrv_CancelInstallAppliance 225
PrlSrv_CancelInstallAppliance function 225
PrlSrv_CheckParallelsServerAlive@PRL_UINT32@PRL_CONST_STR@PRL_BOOL_PTR@PRL_HANDLE_PTR 226
PrlSrv_CommonPrefsBeginEdit 226
PrlSrv_CommonPrefsBeginEdit function 226
PrlSrv_CommonPrefsCommit 227
PrlSrv_CommonPrefsCommit function 227
PrlSrv_CommonPrefsCommitEx 228
PrlSrv_CommonPrefsCommitEx function 228
PrlSrv_ConfigureGenericPci 228
PrlSrv_ConfigureGenericPci function 228
PrlSrv_CopyCtTemplate 229
PrlSrv_CopyCtTemplate function 229
PrlSrv_Create 230
PrlSrv_Create function 230
PrlSrv_CreateUnattendedCd 231
PrlSrv_CreateUnattendedCd function 231

PrlSrv_CreateVm 232
PrlSrv_CreateVm function 232
PrlSrv_CreateVmBackup 233
PrlSrv_CreateVmBackup function 233
PrlSrv_DeleteOfflineService 234
PrlSrv_DeleteOfflineService function 234
PrlSrv_DeleteVirtualNetwork 235
PrlSrv_DeleteVirtualNetwork function 235
PrlSrv_DisableConfirmationMode 236
PrlSrv_DisableConfirmationMode function 236
PrlSrv_EnableConfirmationMode 236
PrlSrv_EnableConfirmationMode function 236
PrlSrv_FsCanCreateFile 237
PrlSrv_FsCanCreateFile function 237
PrlSrv_FsCreateDir 238
PrlSrv_FsCreateDir function 238
PrlSrv_FsGenerateEntryName 238
PrlSrv_FsGenerateEntryName function 238
PrlSrv_FsGetDirEntries 239
PrlSrv_FsGetDirEntries function 239
PrlSrv_FsGetDiskList 240
PrlSrv_FsGetDiskList function 240
PrlSrv_FsRemoveEntry 241
PrlSrv_FsRemoveEntry function 241
PrlSrv_FsRenameEntry 242
PrlSrv_FsRenameEntry function 242
PrlSrv_GetBackupTree 242
PrlSrv_GetBackupTree function 242
PrlSrv_GetCommonPrefs 244
PrlSrv_GetCommonPrefs function 244
PrlSrv_GetCPUPoolsList 245
PrlSrv_GetCPUPoolsList function 245
PrlSrv_GetCtTemplateList 246
PrlSrv_GetCtTemplateList function 246
PrlSrv_GetDefaultVmConfig 246
PrlSrv_GetDefaultVmConfig function 246
PrlSrv_GetDiskFreeSpace 247
PrlSrv_GetDiskFreeSpace function 247
PrlSrv_GetIPPrivateNetworksList 248
PrlSrv_GetIPPrivateNetworksList function 248
PrlSrv_GetLicenseInfo 249
PrlSrv_GetLicenseInfo function 249
PrlSrv_GetNetServiceStatus 249
PrlSrv_GetNetServiceStatus function 249
PrlSrv_GetNetworkClassesList 250
PrlSrv_GetNetworkClassesList function 250
PrlSrv_GetNetworkShapingConfig 251
PrlSrv_GetNetworkShapingConfig function 251
PrlSrv_GetOfflineServicesList 252
PrlSrv_GetOfflineServicesList function 252
PrlSrv_GetPackedProblemReport 252
PrlSrv_GetPackedProblemReport function 252
PrlSrv_GetPerfStats 253
PrlSrv_GetPerfStats function 253
PrlSrv_GetPluginsList 254
PrlSrv_GetPluginsList function 254
PrlSrv_GetProblemReport 254
PrlSrv_GetProblemReport function 254
PrlSrv_GetQuestions 255
PrlSrv_GetQuestions function 255
PrlSrv_GetRestrictionInfo 256
PrlSrv_GetRestrictionInfo function 256
PrlSrv_GetServerInfo 256
PrlSrv_GetServerInfo function 256
PrlSrv_GetSrvConfig 257
PrlSrv_GetSrvConfig function 257
PrlSrv_GetStatistics 258
PrlSrv_GetStatistics function 258
PrlSrv_GetSupportedOses 259
PrlSrv_GetSupportedOses function 259
PrlSrv_GetUserInfo 260
PrlSrv_GetUserInfo function 260
PrlSrv_GetUserInfoList 261
PrlSrv_GetUserInfoList function 261
PrlSrv_GetUserProfile 262
PrlSrv_GetUserProfile function 262
PrlSrv_GetVcmmdConfig 262
PrlSrv_GetVcmmdConfig function 262

PrlSrv_GetVirtualNetworkList 263
PrlSrv_GetVirtualNetworkList function 263
PrlSrv_GetVmConfig 264
PrlSrv_GetVmConfig function 264
PrlSrv_GetVmList 264
PrlSrv_GetVmList function 264
PrlSrv_GetVmListEx 266
PrlSrv_GetVmListEx function 266
PrlSrv_HasRestriction 266
PrlSrv_HasRestriction function 266
PrlSrv_InstallAppliance 267
PrlSrv_InstallAppliance function 267
PrlSrv_IsConfirmationModeEnabled 268
PrlSrv_IsConfirmationModeEnabled function 268
PrlSrv_IsConnected 268
PrlSrv_IsConnected function 268
PrlSrv_IsFeatureSupported 269
PrlSrv_IsFeatureSupported function 269
PrlSrv_IsNonInteractiveSession 270
PrlSrv_IsNonInteractiveSession function 270
PrlSrv_IsProxyConnection 270
PrlSrv_IsProxyConnection function 270
PrlSrv_JoinCPUPool 271
PrlSrv_JoinCPUPool function 271
PrlSrv_LeaveCPUPool 271
PrlSrv_LeaveCPUPool function 271
PrlSrv_Login 272
PrlSrv_Login function 272
PrlSrv_LoginEx 276
PrlSrv_LoginEx function 276
PrlSrv_LoginLocal 276
PrlSrv_LoginLocal function 276
PrlSrv_LoginLocalEx 278
PrlSrv_LoginLocalEx function 278
PrlSrv_Logoff 278
PrlSrv_Logoff function 278
PrlSrv_LookupParallelsServers@PRL_UINT32@PRL_EVENT_HANDLER_PTR@PRL_VOID_PTR 279
PrlSrv_MoveToCPUPool 280
PrlSrv_MoveToCPUPool function 280
PrlSrv_NetServiceRestart 281
PrlSrv_NetServiceRestart function 281
PrlSrv_NetServiceRestoreDefaults 281
PrlSrv_NetServiceRestoreDefaults function 281
PrlSrv_NetServiceStart 282
PrlSrv_NetServiceStart function 282
PrlSrv_NetServiceStop 282
PrlSrv_NetServiceStop function 282
PrlSrv_RecalculateCPUPool 283
PrlSrv_RecalculateCPUPool function 283
PrlSrv_RefreshPlugins 283
PrlSrv_RefreshPlugins function 283
PrlSrv_RegEventHandler 284
PrlSrv_RegEventHandler function 284
PrlSrv_Register3rdPartyVm 285
PrlSrv_Register3rdPartyVm function 285
PrlSrv_RegisterVm 286
PrlSrv_RegisterVm function 286
PrlSrv_RegisterVmEx 288
PrlSrv_RegisterVmEx function 288
PrlSrv_RegisterVmWithUuid 289
PrlSrv_RegisterVmWithUuid function 289
PrlSrv_RemoveCtTemplate 290
PrlSrv_RemoveCtTemplate function 290
PrlSrv_RemoveIPPrivateNetwork 291
PrlSrv_RemoveIPPrivateNetwork function 291
PrlSrv_RemoveVmBackup 292
PrlSrv_RemoveVmBackup function 292
PrlSrv_RestoreVmBackup 293
PrlSrv_RestoreVmBackup function 293
PrlSrv_SendAnswer 294
PrlSrv_SendAnswer function 294
PrlSrv_SendProblemReport 295
PrlSrv_SendProblemReport function 295
PrlSrv_SetNonInteractiveSession 296
PrlSrv_SetNonInteractiveSession function 296
PrlSrv_SetVcmmdConfig 296
PrlSrv_SetVcmmdConfig function 296

PrlSrv_SetVNCEncryption 297
PrlSrv_SetVNCEncryption function 297
PrlSrv_Shutdown 297
PrlSrv_Shutdown function 297
PrlSrv_ShutdownEx 298
PrlSrv_ShutdownEx function 298
PrlSrv_StartSearchVms 299
PrlSrv_StartSearchVms function 299
PrlSrv_StopInstallAppliance 300
PrlSrv_StopInstallAppliance function 300
PrlSrv_SubscribeToHostStatistics 301
PrlSrv_SubscribeToHostStatistics function 301
PrlSrv_SubscribeToPerfStats 305
PrlSrv_SubscribeToPerfStats function 305
PrlSrv_UnregEventHandler 306
PrlSrv_UnregEventHandler function 306
PrlSrv_UnsubscribeFromHostStatistics 307
PrlSrv_UnsubscribeFromHostStatistics function 307
PrlSrv_UnsubscribeFromPerfStats 307
PrlSrv_UnsubscribeFromPerfStats function 307
PrlSrv_UpdateIPPrivateNetwork 308
PrlSrv_UpdateIPPrivateNetwork function 308
PrlSrv_UpdateLicense 309
PrlSrv_UpdateLicense function 309
PrlSrv_UpdateLicenseEx 310
PrlSrv_UpdateLicenseEx function 310
PrlSrv_UpdateNetworkClassesList 310
PrlSrv_UpdateNetworkClassesList function 310
PrlSrv_UpdateNetworkShapingConfig 311
PrlSrv_UpdateNetworkShapingConfig function 311
PrlSrv_UpdateOfflineService 311
PrlSrv_UpdateOfflineService function 311
PrlSrv_UpdateVirtualNetwork 312
PrlSrv_UpdateVirtualNetwork function 312
PrlSrv_UserProfileBeginEdit 313
PrlSrv_UserProfileBeginEdit function 313
PrlSrv_UserProfileCommit 314
PrlSrv_UserProfileCommit function 314
PrlSrvCfg_GetCpuCount 316
PrlSrvCfg_GetCpuCount function 316
PrlSrvCfg_GetCpuFeatures 317
PrlSrvCfg_GetCpuFeatures function 317
PrlSrvCfg_GetCpuFeaturesEx 317
PrlSrvCfg_GetCpuFeaturesEx function 317
PrlSrvCfg_GetCpuFeaturesMaskingCapabilities 318
PrlSrvCfg_GetCpuFeaturesMaskingCapabilities function 318
PrlSrvCfg_GetCpuHvt 318
PrlSrvCfg_GetCpuHvt function 318
PrlSrvCfg_GetCpuMode 319
PrlSrvCfg_GetCpuMode function 319
PrlSrvCfg_GetCpuModel 319
PrlSrvCfg_GetCpuModel function 319
PrlSrvCfg_GetCpuSpeed 320
PrlSrvCfg_GetCpuSpeed function 320
PrlSrvCfg_GetDefaultGateway 320
PrlSrvCfg_GetDefaultGateway function 320
PrlSrvCfg_GetDefaultGatewayIPv6 321
PrlSrvCfg_GetDefaultGatewayIPv6 function 321
PrlSrvCfg_GetDnsServers 322
PrlSrvCfg_GetDnsServers function 322
PrlSrvCfg_GetFloppyDisk 322
PrlSrvCfg_GetFloppyDisk function 322
PrlSrvCfg_GetFloppyDisksCount 323
PrlSrvCfg_GetFloppyDisksCount function 323
PrlSrvCfg_GetGenericPciDevice 323
PrlSrvCfg_GetGenericPciDevice function 323
PrlSrvCfg_GetGenericPciDevicesCount 324
PrlSrvCfg_GetGenericPciDevicesCount function 324
PrlSrvCfg_GetGenericScsiDevice 325
PrlSrvCfg_GetGenericScsiDevice function 325
PrlSrvCfg_GetGenericScsiDevicesCount 325
PrlSrvCfg_GetGenericScsiDevicesCount function 325
PrlSrvCfg_GetHardDisk 326
PrlSrvCfg_GetHardDisk function 326
PrlSrvCfg_GetHardDisksCount 327
PrlSrvCfg_GetHardDisksCount function 327
PrlSrvCfg_GetHostname 327
PrlSrvCfg_GetHostname function 327

PrlSrvCfg_GetHostOsMajor 328
PrlSrvCfg_GetHostOsMajor function 328
PrlSrvCfg_GetHostOsMinor 328
PrlSrvCfg_GetHostOsMinor function 328
PrlSrvCfg_GetHostOsStrPresentation 329
PrlSrvCfg_GetHostOsStrPresentation function 329
PrlSrvCfg_GetHostOsSubMinor 329
PrlSrvCfg_GetHostOsSubMinor function 329
PrlSrvCfg_GetHostOsType 330
PrlSrvCfg_GetHostOsType function 330
PrlSrvCfg_GetHostRamSize 331
PrlSrvCfg_GetHostRamSize function 331
PrlSrvCfg_GetMaxHostNetAdapters 331
PrlSrvCfg_GetMaxHostNetAdapters function 331
PrlSrvCfg_GetMaxVmNetAdapters 332
PrlSrvCfg_GetMaxVmNetAdapters function 332
PrlSrvCfg_GetNetAdapter 332
PrlSrvCfg_GetNetAdapter function 332
PrlSrvCfg_GetNetAdaptersCount 333
PrlSrvCfg_GetNetAdaptersCount function 333
PrlSrvCfg_GetOpticalDisk 333
PrlSrvCfg_GetOpticalDisk function 333
PrlSrvCfg_GetOpticalDisksCount 334
PrlSrvCfg_GetOpticalDisksCount function 334
PrlSrvCfg_GetParallelPort 334
PrlSrvCfg_GetParallelPort function 334
PrlSrvCfg_GetParallelPortsCount 335
PrlSrvCfg_GetParallelPortsCount function 335
PrlSrvCfg_GetPrinter 336
PrlSrvCfg_GetPrinter function 336
PrlSrvCfg_GetPrintersCount 336
PrlSrvCfg_GetPrintersCount function 336
PrlSrvCfg_GetSearchDomains 337
PrlSrvCfg_GetSearchDomains function 337
PrlSrvCfg_GetSerialPort 337
PrlSrvCfg_GetSerialPort function 337
PrlSrvCfg_GetSerialPortsCount 338
PrlSrvCfg_GetSerialPortsCount function 338
PrlSrvCfg_GetSoundMixerDev 338
PrlSrvCfg_GetSoundMixerDev function 338
PrlSrvCfg_GetSoundMixerDevsCount 339
PrlSrvCfg_GetSoundMixerDevsCount function 339
PrlSrvCfg_GetSoundOutputDev 340
PrlSrvCfg_GetSoundOutputDev function 340
PrlSrvCfg_GetSoundOutputDevsCount 340
PrlSrvCfg_GetSoundOutputDevsCount function 340
PrlSrvCfg_GetUsbDev 341
PrlSrvCfg_GetUsbDev function 341
PrlSrvCfg_GetUsbDevsCount 341
PrlSrvCfg_GetUsbDevsCount function 341
PrlSrvCfg_IsSoundDefaultEnabled 342
PrlSrvCfg_IsSoundDefaultEnabled function 342
PrlSrvCfg_IsUsbSupported 342
PrlSrvCfg_IsUsbSupported function 342
PrlSrvCfg_IsVtdSupported 343
PrlSrvCfg_IsVtdSupported function 343
PrlSrvCfgDev_GetDeviceState 112
PrlSrvCfgDev_GetDeviceState function 112
PrlSrvCfgDev_GetId 112
PrlSrvCfgDev_GetId function 112
PrlSrvCfgDev_GetName 113
PrlSrvCfgDev_GetName function 113
PrlSrvCfgDev_GetType 114
PrlSrvCfgDev_GetType function 114
PrlSrvCfgDev_IsConnectedToVm 114
PrlSrvCfgDev_IsConnectedToVm function 114
PrlSrvCfgDev_SetDeviceState 115
PrlSrvCfgDev_SetDeviceState function 115
PrlSrvCfgHdd_GetDevId 117
PrlSrvCfgHdd_GetDevId function 117
PrlSrvCfgHdd_GetDevName 118
PrlSrvCfgHdd_GetDevName function 118
PrlSrvCfgHdd_GetDevSize 118
PrlSrvCfgHdd_GetDevSize function 118
PrlSrvCfgHdd_GetDiskIndex 119
PrlSrvCfgHdd_GetDiskIndex function 119
PrlSrvCfgHdd_GetPart 119
PrlSrvCfgHdd_GetPart function 119

PrlSrvCfgHdd_GetPartsCount 120
PrlSrvCfgHdd_GetPartsCount function 120
PrlSrvCfgHddPart_GetIndex 121
PrlSrvCfgHddPart_GetIndex function 121
PrlSrvCfgHddPart_GetName 122
PrlSrvCfgHddPart_GetName function 122
PrlSrvCfgHddPart_GetSize 122
PrlSrvCfgHddPart_GetSize function 122
PrlSrvCfgHddPart_GetSysName 123
PrlSrvCfgHddPart_GetSysName function 123
PrlSrvCfgHddPart_GetType 124
PrlSrvCfgHddPart_GetType function 124
PrlSrvCfgHddPart_IsActive 124
PrlSrvCfgHddPart_IsActive function 124
PrlSrvCfgHddPart_IsInUse 125
PrlSrvCfgHddPart_IsInUse function 125
PrlSrvCfgHddPart_IsLogical 125
PrlSrvCfgHddPart_IsLogical function 125
PrlSrvCfgNet_GetDefaultGateway 126
PrlSrvCfgNet_GetDefaultGateway function 126
PrlSrvCfgNet_GetDefaultGatewayIPv6 127
PrlSrvCfgNet_GetDefaultGatewayIPv6 function 127
PrlSrvCfgNet_GetDnsServers 128
PrlSrvCfgNet_GetDnsServers function 128
PrlSrvCfgNet_GetMacAddress 128
PrlSrvCfgNet_GetMacAddress function 128
PrlSrvCfgNet_GetNetAdapterType 129
PrlSrvCfgNet_GetNetAdapterType function 129
PrlSrvCfgNet_GetNetAddresses 129
PrlSrvCfgNet_GetNetAddresses function 129
PrlSrvCfgNet_GetSearchDomains 130
PrlSrvCfgNet_GetSearchDomains function 130
PrlSrvCfgNet_GetSysIndex 131
PrlSrvCfgNet_GetSysIndex function 131
PrlSrvCfgNet_GetVlanTag 131
PrlSrvCfgNet_GetVlanTag function 131
PrlSrvCfgNet_IsConfigureWithDhcp 132
PrlSrvCfgNet_IsConfigureWithDhcp function 132
PrlSrvCfgNet_IsConfigureWithDhcpIPv6 132
PrlSrvCfgNet_IsConfigureWithDhcpIPv6 function 132
PrlSrvCfgNet_IsEnabled 133
PrlSrvCfgNet_IsEnabled function 133
PrlSrvCfgPci_GetDeviceClass 116
PrlSrvCfgPci_GetDeviceClass function 116
PrlSrvCfgPci_IsPrimaryDevice 343
PrlSrvCfgPci_IsPrimaryDevice function 343
PrlSrvInfo_GetApplicationMode 345
PrlSrvInfo_GetApplicationMode function 345
PrlSrvInfo_GetCmdPort 345
PrlSrvInfo_GetCmdPort function 345
PrlSrvInfo_GetHostName 346
PrlSrvInfo_GetHostName function 346
PrlSrvInfo_GetOsVersion 346
PrlSrvInfo_GetOsVersion function 346
PrlSrvInfo_GetProductVersion 347
PrlSrvInfo_GetProductVersion function 347
PrlSrvInfo_GetServerUuid 348
PrlSrvInfo_GetServerUuid function 348
PrlSrvInfo_GetStartTime 348
PrlSrvInfo_GetStartTime function 348
PrlSrvInfo_GetStartTimeMonotonic 349
PrlSrvInfo_GetStartTimeMonotonic function 349
PrlStat_GetCpusStatsCount 361
PrlStat_GetCpusStatsCount function 361
PrlStat_GetCpuStat 362
PrlStat_GetCpuStat function 362
PrlStat_GetFreeRamSize 362
PrlStat_GetFreeRamSize function 362
PrlStat_GetOsUptime 363
PrlStat_GetOsUptime function 363
PrlStat_GetRealRamSize 363
PrlStat_GetRealRamSize function 363
PrlStat_GetTotalRamSize 364
PrlStat_GetTotalRamSize function 364
PrlStat_GetUsageRamSize 364
PrlStat_GetUsageRamSize function 364
PrlStatCpu_GetCpuUsage 365
PrlStatCpu_GetCpuUsage function 365

PrlStatCpu_GetSystemTime 366
PrlStatCpu_GetSystemTime function 366
PrlStatCpu_GetTotalTime 366
PrlStatCpu_GetTotalTime function 366
PrlStatCpu_GetUserTime 367
PrlStatCpu_GetUserTime function 367
PrlStrList_AddItem 358
PrlStrList_AddItem function 358
PrlStrList_GetItem 358
PrlStrList_GetItem function 358
PrlStrList_GetItemsCount 359
PrlStrList_GetItemsCount function 359
PrlStrList_RemoveItem 360
PrlStrList_RemoveItem function 360
PrlUsbIdent_GetFriendlyName 369
PrlUsbIdent_GetFriendlyName function 369
PrlUsbIdent_GetSystemName 369
PrlUsbIdent_GetSystemName function 369
PrlUsbIdent_GetVmUuidAssociation 370
PrlUsbIdent_GetVmUuidAssociation function 370
PrlUsrCfg_CanChangeSrvSets 375
PrlUsrCfg_CanChangeSrvSets function 375
PrlUsrCfg_CanUseMngConsole 376
PrlUsrCfg_CanUseMngConsole function 376
PrlUsrCfg_GetDefaultVmFolder 376
PrlUsrCfg_GetDefaultVmFolder function 376
PrlUsrCfg_GetVmDirUuid 377
PrlUsrCfg_GetVmDirUuid function 377
PrlUsrCfg_IsLocalAdministrator 378
PrlUsrCfg_IsLocalAdministrator function 378
PrlUsrCfg_SetDefaultVmFolder 378
PrlUsrCfg_SetDefaultVmFolder function 378
PrlUsrCfg_SetVmDirUuid 379
PrlUsrCfg_SetVmDirUuid function 379
PrlUsrInfo_CanChangeSrvSets 371
PrlUsrInfo_CanChangeSrvSets function 371
PrlUsrInfo_GetDefaultVmFolder 372
PrlUsrInfo_GetDefaultVmFolder function 372
PrlUsrInfo_GetName 372
PrlUsrInfo_GetName function 372
PrlUsrInfo_GetSessionCount 373
PrlUsrInfo_GetSessionCount function 373
PrlUsrInfo_GetUuid 374
PrlUsrInfo_GetUuid function 374
PrlVcmmdConfig_GetPolicy 380
PrlVcmmdConfig_GetPolicy function 380
PrlVcmmdConfig_SetPolicy 380
PrlVcmmdConfig_SetPolicy function 380
PrlVirtNet_Create 508
PrlVirtNet_Create function 508
PrlVirtNet_GetAdapterIndex 509
PrlVirtNet_GetAdapterIndex function 509
PrlVirtNet_GetAdapterName 510
PrlVirtNet_GetAdapterName function 510
PrlVirtNet_GetBoundAdapterInfo 510
PrlVirtNet_GetBoundAdapterInfo function 510
PrlVirtNet_GetBoundCardMac 511
PrlVirtNet_GetBoundCardMac function 511
PrlVirtNet_GetDescription 512
PrlVirtNet_GetDescription function 512
PrlVirtNet_GetDhcpIP6Address 512
PrlVirtNet_GetDhcpIP6Address function 512
PrlVirtNet_GetDhcpIPAddress 513
PrlVirtNet_GetDhcpIPAddress function 513
PrlVirtNet_GetHostIP6Address 514
PrlVirtNet_GetHostIP6Address function 514
PrlVirtNet_GetHostIPAddress 514
PrlVirtNet_GetHostIPAddress function 514
PrlVirtNet_GetIP6NetMask 515
PrlVirtNet_GetIP6NetMask function 515
PrlVirtNet_GetIP6ScopeEnd 516
PrlVirtNet_GetIP6ScopeEnd function 516
PrlVirtNet_GetIP6ScopeStart 516
PrlVirtNet_GetIP6ScopeStart function 516
PrlVirtNet_GetIPNetMask 517
PrlVirtNet_GetIPNetMask function 517
PrlVirtNet_GetIPScopeEnd 518
PrlVirtNet_GetIPScopeEnd function 518

PrlVirtNet_GetIPScopeStart 518
PrlVirtNet_GetIPScopeStart function 518
PrlVirtNet_GetNetworkId 519
PrlVirtNet_GetNetworkId function 519
PrlVirtNet_GetNetworkType 520
PrlVirtNet_GetNetworkType function 520
PrlVirtNet_GetPortForwardList 520
PrlVirtNet_GetPortForwardList function 520
PrlVirtNet_GetVlanTag 521
PrlVirtNet_GetVlanTag function 521
PrlVirtNet_IsAdapterEnabled 521
PrlVirtNet_IsAdapterEnabled function 521
PrlVirtNet_IsDHCP6ServerEnabled 522
PrlVirtNet_IsDHCP6ServerEnabled function 522
PrlVirtNet_IsDHCPServerEnabled 522
PrlVirtNet_IsDHCPServerEnabled function 522
PrlVirtNet_IsEnabled 523
PrlVirtNet_IsEnabled function 523
PrlVirtNet_IsNATServerEnabled 524
PrlVirtNet_IsNATServerEnabled function 524
PrlVirtNet_SetAdapterEnabled 524
PrlVirtNet_SetAdapterEnabled function 524
PrlVirtNet_SetAdapterIndex 525
PrlVirtNet_SetAdapterIndex function 525
PrlVirtNet_SetAdapterName 525
PrlVirtNet_SetAdapterName function 525
PrlVirtNet_SetBoundCardMac 526
PrlVirtNet_SetBoundCardMac function 526
PrlVirtNet_SetDescription 526
PrlVirtNet_SetDescription function 526
PrlVirtNet_SetDHCP6ServerEnabled 527
PrlVirtNet_SetDHCP6ServerEnabled function 527
PrlVirtNet_SetDhcpIP6Address 527
PrlVirtNet_SetDhcpIP6Address function 527
PrlVirtNet_SetDhcpIPAddress 528
PrlVirtNet_SetDhcpIPAddress function 528
PrlVirtNet_SetDHCPServerEnabled 528
PrlVirtNet_SetDHCPServerEnabled function 528
PrlVirtNet_SetEnabled 529
PrlVirtNet_SetEnabled function 529
PrlVirtNet_SetHostIP6Address 529
PrlVirtNet_SetHostIP6Address function 529
PrlVirtNet_SetHostIPAddress 530
PrlVirtNet_SetHostIPAddress function 530
PrlVirtNet_SetIP6NetMask 531
PrlVirtNet_SetIP6NetMask function 531
PrlVirtNet_SetIP6ScopeEnd 531
PrlVirtNet_SetIP6ScopeEnd function 531
PrlVirtNet_SetIP6ScopeStart 532
PrlVirtNet_SetIP6ScopeStart function 532
PrlVirtNet_SetIPNetMask 532
PrlVirtNet_SetIPNetMask function 532
PrlVirtNet_SetIPScopeEnd 533
PrlVirtNet_SetIPScopeEnd function 533
PrlVirtNet_SetIPScopeStart 533
PrlVirtNet_SetIPScopeStart function 533
PrlVirtNet_SetNATServerEnabled 534
PrlVirtNet_SetNATServerEnabled function 534
PrlVirtNet_SetNetworkId 534
PrlVirtNet_SetNetworkId function 534
PrlVirtNet_SetNetworkType 535
PrlVirtNet_SetNetworkType function 535
PrlVirtNet_SetPortForwardList 535
PrlVirtNet_SetPortForwardList function 535
PrlVirtNet_SetVlanTag 536
PrlVirtNet_SetVlanTag function 536
PrlVm_AuthWithGuestSecurityDb 446
PrlVm_AuthWithGuestSecurityDb function 446
PrlVm_BeginEdit 446
PrlVm_BeginEdit function 446
PrlVm_CancelCompact 448
PrlVm_CancelCompact function 448
PrlVm_CancelConvertDisks 448
PrlVm_CancelConvertDisks function 448
PrlVm_CaptureScreen 449
PrlVm_CaptureScreen function 449
PrlVm_ChangeSid 450
PrlVm_ChangeSid function 450

PrlVm_Clone 450
PrlVm_Clone function 450
PrlVm_CloneEx 451
PrlVm_CloneEx function 451
PrlVm_CloneWithUuid 452
PrlVm_CloneWithUuid function 452
PrlVm_Commit 453
PrlVm_Commit function 453
PrlVm_CommitEncryption 454
PrlVm_CommitEncryption function 454
PrlVm_CommitEx 455
PrlVm_CommitEx function 455
PrlVm_Compact 456
PrlVm_Compact function 456
PrlVm_Connect 456
PrlVm_Connect function 456
PrlVm_ConvertDisks 457
PrlVm_ConvertDisks function 457
PrlVm_CreateEvent 458
PrlVm_CreateEvent function 458
PrlVm_CreateSnapshot 458
PrlVm_CreateSnapshot function 458
PrlVm_CreateSnapshotEx 459
PrlVm_CreateSnapshotEx function 459
PrlVm_CreateUnattendedFloppy 460
PrlVm_CreateUnattendedFloppy function 460
PrlVm_Delete 461
PrlVm_Delete function 461
PrlVm_DeleteSnapshot 461
PrlVm_DeleteSnapshot function 461
PrlVm_Disconnect 462
PrlVm_Disconnect function 462
PrlVm_DropSuspendedState 463
PrlVm_DropSuspendedState function 463
PrlVm_GenerateVmDevFilename 463
PrlVm_GenerateVmDevFilename function 463
PrlVm_GetConfig 464
PrlVm_GetConfig function 464
PrlVm_GetPackedProblemReport 465
PrlVm_GetPackedProblemReport function 465
PrlVm_GetPerfStats 465
PrlVm_GetPerfStats function 465
PrlVm_GetProblemReport 466
PrlVm_GetProblemReport function 466
PrlVm_GetQuestions 467
PrlVm_GetQuestions function 467
PrlVm_GetSnapshotsTree 467
PrlVm_GetSnapshotsTree function 467
PrlVm_GetSnapshotsTreeEx 468
PrlVm_GetSnapshotsTreeEx function 468
PrlVm_GetState 468
PrlVm_GetState function 468
PrlVm_GetStatistics 469
PrlVm_GetStatistics function 469
PrlVm_GetStatisticsEx 470
PrlVm_GetStatisticsEx function 470
PrlVm_GetSuspendedScreen 470
PrlVm_GetSuspendedScreen function 470
PrlVm_GetToolsState 471
PrlVm_GetToolsState function 471
PrlVm_InitiateDevStateNotifications 472
PrlVm_InitiateDevStateNotifications function 472
PrlVm_InstallTools 472
PrlVm_InstallTools function 472
PrlVm_Lock 473
PrlVm_Lock function 473
PrlVm_LoginInGuest 473
PrlVm_LoginInGuest function 473
PrlVm_Migrate 475
PrlVm_Migrate function 475
PrlVm_MigrateCancel 476
PrlVm_MigrateCancel function 476
PrlVm_MigrateEx 476
PrlVm_MigrateEx function 476
PrlVm_MigrateWithRename 477
PrlVm_MigrateWithRename function 477
PrlVm_MigrateWithRenameEx 478
PrlVm_MigrateWithRenameEx function 478

PrlVm_Mount 479
PrlVm_Mount function 479
PrlVm_Move 480
PrlVm_Move function 480
PrlVm_Pause 481
PrlVm_Pause function 481
PrlVm_RefreshConfig 481
PrlVm_RefreshConfig function 481
PrlVm_RefreshConfigEx 482
PrlVm_RefreshConfigEx function 482
PrlVm_Reg 483
PrlVm_Reg function 483
PrlVm_RegEventHandler 485
PrlVm_RegEventHandler function 485
PrlVm_RegEx 485
PrlVm_RegEx function 485
PrlVm_Reset 486
PrlVm_Reset function 486
PrlVm_ResetUptime 487
PrlVm_ResetUptime function 487
PrlVm_Restart 488
PrlVm_Restart function 488
PrlVm_Restore 488
PrlVm_Restore function 488
PrlVm_Resume 489
PrlVm_Resume function 489
PrlVm_SetConfig 489
PrlVm_SetConfig function 489
PrlVm_SetUserPasswd 490
PrlVm_SetUserPasswd function 490
PrlVm_Start 491
PrlVm_Start function 491
PrlVm_StartEx 491
PrlVm_StartEx function 491
PrlVm_StartVncServer 492
PrlVm_StartVncServer function 492
PrlVm_Stop 493
PrlVm_Stop function 493
PrlVm_StopEx 493
PrlVm_StopEx function 493
PrlVm_StopVncServer 494
PrlVm_StopVncServer function 494
PrlVm_SubscribeToGuestStatistics 495
PrlVm_SubscribeToGuestStatistics function 495
PrlVm_SubscribeToPerfStats 495
PrlVm_SubscribeToPerfStats function 495
PrlVm_Suspend 496
PrlVm_Suspend function 496
PrlVm_SwitchToSnapshot 497
PrlVm_SwitchToSnapshot function 497
PrlVm_SwitchToSnapshotEx 498
PrlVm_SwitchToSnapshotEx function 498
PrlVm_Umount 498
PrlVm_Umount function 498
PrlVm_Unlock 499
PrlVm_Unlock function 499
PrlVm_Unreg 500
PrlVm_Unreg function 500
PrlVm_UnregEventHandler 500
PrlVm_UnregEventHandler function 500
PrlVm_UnsubscribeFromGuestStatistics 501
PrlVm_UnsubscribeFromGuestStatistics function 501
PrlVm_UnsubscribeFromPerfStats 501
PrlVm_UnsubscribeFromPerfStats function 501
PrlVm_UpdateSecurity 502
PrlVm_UpdateSecurity function 502
PrlVm_UpdateSnapshotData 503
PrlVm_UpdateSnapshotData function 503
PrlVm_ValidateConfig 504
PrlVm_ValidateConfig function 504
PrlVmCfg_AddDefaultDevice 547
PrlVmCfg_AddDefaultDevice function 547
PrlVmCfg_AddDefaultDeviceEx 548
PrlVmCfg_AddDefaultDeviceEx function 548
PrlVmCfg_ApplyConfigSample 549
PrlVmCfg_ApplyConfigSample function 549
PrlVmCfg_CreateBootDev 550
PrlVmCfg_CreateBootDev function 550

PrlVmCfg_CreateShare 550
PrlVmCfg_CreateShare function 550
PrlVmCfg_CreateVmDev 551
PrlVmCfg_CreateVmDev function 551
PrlVmCfg_Get3DAccelerationMode 552
PrlVmCfg_Get3DAccelerationMode function 552
PrlVmCfg_GetAccessRights 552
PrlVmCfg_GetAccessRights function 552
PrlVmCfg_GetActionOnGuestCrash 553
PrlVmCfg_GetActionOnGuestCrash function 553
PrlVmCfg_GetActionOnStopMode 553
PrlVmCfg_GetActionOnStopMode function 553
PrlVmCfg_GetActionOnWindowClose 554
PrlVmCfg_GetActionOnWindowClose function 554
PrlVmCfg_GetAllDevices 554
PrlVmCfg_GetAllDevices function 554
PrlVmCfg_GetAppTemplateList 555
PrlVmCfg_GetAppTemplateList function 555
PrlVmCfg_GetAutoCompressInterval 555
PrlVmCfg_GetAutoCompressInterval function 555
PrlVmCfg_GetAutoStart 556
PrlVmCfg_GetAutoStart function 556
PrlVmCfg_GetAutoStartDelay 557
PrlVmCfg_GetAutoStartDelay function 557
PrlVmCfg_GetAutoStop 557
PrlVmCfg_GetAutoStop function 557
PrlVmCfg_GetBackgroundPriority 558
PrlVmCfg_GetBackgroundPriority function 558
PrlVmCfg_GetBootDev 558
PrlVmCfg_GetBootDev function 558
PrlVmCfg_GetBootDevCount 559
PrlVmCfg_GetBootDevCount function 559
PrlVmCfg_GetCapabilitiesMask 559
PrlVmCfg_GetCapabilitiesMask function 559
PrlVmCfg_GetConfigValidity 560
PrlVmCfg_GetConfigValidity function 560
PrlVmCfg_GetConfirmationsList 561
PrlVmCfg_GetConfirmationsList function 561
PrlVmCfg_GetCpuAccelLevel 561
PrlVmCfg_GetCpuAccelLevel function 561
PrlVmCfg_GetCpuCount 562
PrlVmCfg_GetCpuCount function 562
PrlVmCfg_GetCpuLimit 562
PrlVmCfg_GetCpuLimit function 562
PrlVmCfg_GetCpuLimitEx 563
PrlVmCfg_GetCpuLimitEx function 563
PrlVmCfg_GetCpuMask 563
PrlVmCfg_GetCpuMask function 563
PrlVmCfg_GetCpuMode 564
PrlVmCfg_GetCpuMode function 564
PrlVmCfg_GetCpuUnits 565
PrlVmCfg_GetCpuUnits function 565
PrlVmCfg_GetCtId 565
PrlVmCfg_GetCtId function 565
PrlVmCfg_GetCustomProperty 566
PrlVmCfg_GetCustomProperty function 566
PrlVmCfg_GetDefaultHddSize 567
PrlVmCfg_GetDefaultHddSize function 567
PrlVmCfg_GetDefaultMemSize 567
PrlVmCfg_GetDefaultMemSize function 567
PrlVmCfg_GetDefaultVideoRamSize 568
PrlVmCfg_GetDefaultVideoRamSize function 568
PrlVmCfg_GetDescription 569
PrlVmCfg_GetDescription function 569
PrlVmCfg_GetDevByType 570
PrlVmCfg_GetDevByType function 570
PrlVmCfg_GetDevsCount 570
PrlVmCfg_GetDevsCount function 570
PrlVmCfg_GetDevsCountByType 571
PrlVmCfg_GetDevsCountByType function 571
PrlVmCfg_GetDisplayDev 572
PrlVmCfg_GetDisplayDev function 572
PrlVmCfg_GetDisplayDevsCount 572
PrlVmCfg_GetDisplayDevsCount function 572
PrlVmCfg_GetDnsServers 573
PrlVmCfg_GetDnsServers function 573
PrlVmCfg_GetExternalBootDevice 573
PrlVmCfg_GetExternalBootDevice function 573

PrlVmCfg_GetFeaturesMask 574
PrlVmCfg_GetFeaturesMask function 574
PrlVmCfg_GetFloppyDisk 575
PrlVmCfg_GetFloppyDisk function 575
PrlVmCfg_GetFloppyDisksCount 575
PrlVmCfg_GetFloppyDisksCount function 575
PrlVmCfg_GetForegroundPriority 576
PrlVmCfg_GetForegroundPriority function 576
PrlVmCfg_GetFreeDiskSpaceRatio 576
PrlVmCfg_GetFreeDiskSpaceRatio function 576
PrlVmCfg_GetGenericPciDev 577
PrlVmCfg_GetGenericPciDev function 577
PrlVmCfg_GetGenericPciDevsCount 578
PrlVmCfg_GetGenericPciDevsCount function 578
PrlVmCfg_GetGenericScsiDev 578
PrlVmCfg_GetGenericScsiDev function 578
PrlVmCfg_GetGenericScsiDevsCount 579
PrlVmCfg_GetGenericScsiDevsCount function 579
PrlVmCfg_GetGuestCpuLimitType 580
PrlVmCfg_GetGuestCpuLimitType function 580
PrlVmCfg_GetHardDisk 580
PrlVmCfg_GetHardDisk function 580
PrlVmCfg_GetHardDisksCount 581
PrlVmCfg_GetHardDisksCount function 581
PrlVmCfg_GetHighAvailabilityPriority 581
PrlVmCfg_GetHighAvailabilityPriority function 581
PrlVmCfg_GetHomePath 582
PrlVmCfg_GetHomePath function 582
PrlVmCfg_GetHostname 582
PrlVmCfg_GetHostname function 582
PrlVmCfg_GetIcon 583
PrlVmCfg_GetIcon function 583
PrlVmCfg_GetIoLimit 584
PrlVmCfg_GetIoLimit function 584
PrlVmCfg_GetIoPriority 584
PrlVmCfg_GetIoPriority function 584
PrlVmCfg_GetIopsLimit 585
PrlVmCfg_GetIopsLimit function 585
PrlVmCfg_GetLastModifiedDate 585
PrlVmCfg_GetLastModifiedDate function 585
PrlVmCfg_GetLastModifierName 586
PrlVmCfg_GetLastModifierName function 586
PrlVmCfg_GetLinkedVmUuid 587
PrlVmCfg_GetLinkedVmUuid function 587
PrlVmCfg_GetLocation 587
PrlVmCfg_GetLocation function 587
PrlVmCfg_GetMemGuaranteeSize 588
PrlVmCfg_GetMemGuaranteeSize function 588
PrlVmCfg_GetName 589
PrlVmCfg_GetName function 589
PrlVmCfg_GetNetAdapter 589
PrlVmCfg_GetNetAdapter function 589
PrlVmCfg_GetNetAdaptersCount 590
PrlVmCfg_GetNetAdaptersCount function 590
PrlVmCfg_GetNetfilterMode 590
PrlVmCfg_GetNetfilterMode function 590
PrlVmCfg_GetNetworkRateList 591
PrlVmCfg_GetNetworkRateList function 591
PrlVmCfg_GetNodeMask 592
PrlVmCfg_GetNodeMask function 592
PrlVmCfg_GetOfflineServices 592
PrlVmCfg_GetOfflineServices function 592
PrlVmCfg_GetOpticalDisk 593
PrlVmCfg_GetOpticalDisk function 593
PrlVmCfg_GetOpticalDisksCount 594
PrlVmCfg_GetOpticalDisksCount function 594
PrlVmCfg_GetOptimizeModifiersMode 594
PrlVmCfg_GetOptimizeModifiersMode function 594
PrlVmCfg_GetOptionsOnGuestCrash 595
PrlVmCfg_GetOptionsOnGuestCrash function 595
PrlVmCfg_GetOsTemplate 595
PrlVmCfg_GetOsTemplate function 595
PrlVmCfg_GetOsType 596
PrlVmCfg_GetOsType function 596
PrlVmCfg_GetOsVersion 596
PrlVmCfg_GetOsVersion function 596
PrlVmCfg_GetParallelPort 597
PrlVmCfg_GetParallelPort function 597

PrlVmCfg_GetParallelPortsCount 598
PrlVmCfg_GetParallelPortsCount function 598
PrlVmCfg_GetProfile 598
PrlVmCfg_GetProfile function 598
PrlVmCfg_GetRamSize 599
PrlVmCfg_GetRamSize function 599
PrlVmCfg_GetResource 599
PrlVmCfg_GetResource function 599
PrlVmCfg_GetSearchDomains 600
PrlVmCfg_GetSearchDomains function 600
PrlVmCfg_GetSerialPort 600
PrlVmCfg_GetSerialPort function 600
PrlVmCfg_GetSerialPortsCount 601
PrlVmCfg_GetSerialPortsCount function 601
PrlVmCfg_GetServerHost 602
PrlVmCfg_GetServerHost function 602
PrlVmCfg_GetServerUuid 602
PrlVmCfg_GetServerUuid function 602
PrlVmCfg_GetShare 603
PrlVmCfg_GetShare function 603
PrlVmCfg_GetSharesCount 604
PrlVmCfg_GetSharesCount function 604
PrlVmCfg_GetSmartGuardInterval 604
PrlVmCfg_GetSmartGuardInterval function 604
PrlVmCfg_GetSmartGuardMaxSnapshotsCount 605
PrlVmCfg_GetSmartGuardMaxSnapshotsCount function 605
PrlVmCfg_GetSoundDev 606
PrlVmCfg_GetSoundDev function 606
PrlVmCfg_GetSoundDevsCount 606
PrlVmCfg_GetSoundDevsCount function 606
PrlVmCfg_GetStartLoginMode 607
PrlVmCfg_GetStartLoginMode function 607
PrlVmCfg_GetStartUserLogin 607
PrlVmCfg_GetStartUserLogin function 607
PrlVmCfg_GetSystemFlags 608
PrlVmCfg_GetSystemFlags function 608
PrlVmCfg_GetTimeSyncInterval 609
PrlVmCfg_GetTimeSyncInterval function 609
PrlVmCfg_GetUndoDisksMode 609
PrlVmCfg_GetUndoDisksMode function 609
PrlVmCfg_GetUptime 610
PrlVmCfg_GetUptime function 610
PrlVmCfg_GetUptimeStartDate 610
PrlVmCfg_GetUptimeStartDate function 610
PrlVmCfg_GetUsbDevice 611
PrlVmCfg_GetUsbDevice function 611
PrlVmCfg_GetUsbDevicesCount 612
PrlVmCfg_GetUsbDevicesCount function 612
PrlVmCfg_GetUuid 612
PrlVmCfg_GetUuid function 612
PrlVmCfg_GetVideoRamSize 613
PrlVmCfg_GetVideoRamSize function 613
PrlVmCfg_GetVmInfo 613
PrlVmCfg_GetVmInfo function 613
PrlVmCfg_GetVmType 614
PrlVmCfg_GetVmType function 614
PrlVmCfg_GetVNCHostName 614
PrlVmCfg_GetVNCHostName function 614
PrlVmCfg_GetVNCMode 615
PrlVmCfg_GetVNCMode function 615
PrlVmCfg_GetVNCPassword 616
PrlVmCfg_GetVNCPassword function 616
PrlVmCfg_GetVNCPort 616
PrlVmCfg_GetVNCPort function 616
PrlVmCfg_GetVNCWebSocketPort 617
PrlVmCfg_GetVNCWebSocketPort function 617
PrlVmCfg_GetWindowMode 617
PrlVmCfg_GetWindowMode function 617
PrlVmCfg_IsAdaptiveHypervisorEnabled 618
PrlVmCfg_IsAdaptiveHypervisorEnabled function 618
PrlVmCfg_IsAllowSelectBootDevice 618
PrlVmCfg_IsAllowSelectBootDevice function 618
PrlVmCfg_IsAutoApplyIpOnly 619
PrlVmCfg_IsAutoApplyIpOnly function 619
PrlVmCfg_IsAutoCompressEnabled 620
PrlVmCfg_IsAutoCompressEnabled function 620
PrlVmCfg_IsBatteryStatusEnabled 620
PrlVmCfg_IsBatteryStatusEnabled function 620

PrlVmCfg_IsCloseAppOnShutdown 621
PrlVmCfg_IsCloseAppOnShutdown function 621
PrlVmCfg_IsConfigInvalid 621
PrlVmCfg_IsConfigInvalid function 621
PrlVmCfg_IsCpuHotplugEnabled 622
PrlVmCfg_IsCpuHotplugEnabled function 622
PrlVmCfg_IsCpuVtxEnabled 622
PrlVmCfg_IsCpuVtxEnabled function 622
PrlVmCfg_IsCustomProfile 623
PrlVmCfg_IsCustomProfile function 623
PrlVmCfg_IsDefaultDeviceNeeded 623
PrlVmCfg_IsDefaultDeviceNeeded function 623
PrlVmCfg_IsDisableAPIC 624
PrlVmCfg_IsDisableAPIC function 624
PrlVmCfg_IsDisableSpeaker 625
PrlVmCfg_IsDisableSpeaker function 625
PrlVmCfg_IsDiskCacheWriteBack 625
PrlVmCfg_IsDiskCacheWriteBack function 625
PrlVmCfg_IsEfiEnabled 626
PrlVmCfg_IsEfiEnabled function 626
PrlVmCfg_IsGuestSharingAutoMount 626
PrlVmCfg_IsGuestSharingAutoMount function 626
PrlVmCfg_IsGuestSharingEnabled 627
PrlVmCfg_IsGuestSharingEnabled function 627
PrlVmCfg_IsGuestSharingEnableSpotlight 628
PrlVmCfg_IsGuestSharingEnableSpotlight function 628
PrlVmCfg_IsHighAvailabilityEnabled 628
PrlVmCfg_IsHighAvailabilityEnabled function 628
PrlVmCfg_IsHighResolutionEnabled 505
PrlVmCfg_IsHighResolutionEnabled function 505
PrlVmCfg_IsHostSharingEnabled 629
PrlVmCfg_IsHostSharingEnabled function 629
PrlVmCfg_IsIsolatedVmEnabled 630
PrlVmCfg_IsIsolatedVmEnabled function 630
PrlVmCfg_IsLockGuestOnSuspendEnabled 630
PrlVmCfg_IsLockGuestOnSuspendEnabled function 630
PrlVmCfg_IsLockInFullScreenMode 631
PrlVmCfg_IsLockInFullScreenMode function 631
PrlVmCfg_IsLongerBatteryLifeEnabled 631
PrlVmCfg_IsLongerBatteryLifeEnabled function 631
PrlVmCfg_IsMapSharedFoldersOnLetters 632
PrlVmCfg_IsMapSharedFoldersOnLetters function 632
PrlVmCfg_IsNestedVirtualizationEnabled 632
PrlVmCfg_IsNestedVirtualizationEnabled function 632
PrlVmCfg_IsOfflineManagementEnabled 633
PrlVmCfg_IsOfflineManagementEnabled function 633
PrlVmCfg_IsOsResInFullScrMode 634
PrlVmCfg_IsOsResInFullScrMode function 634
PrlVmCfg_IsPMUVirtualizationEnabled 634
PrlVmCfg_IsPMUVirtualizationEnabled function 634
PrlVmCfg_IsRamHotplugEnabled 635
PrlVmCfg_IsRamHotplugEnabled function 635
PrlVmCfg_IsRateBound 635
PrlVmCfg_IsRateBound function 635
PrlVmCfg_IsScrResEnabled 636
PrlVmCfg_IsScrResEnabled function 636
PrlVmCfg_IsShareAllHostDisks 636
PrlVmCfg_IsShareAllHostDisks function 636
PrlVmCfg_IsShareClipboard 637
PrlVmCfg_IsShareClipboard function 637
PrlVmCfg_IsSharedProfileEnabled 637
PrlVmCfg_IsSharedProfileEnabled function 637
PrlVmCfg_IsShareUserHomeDir 638
PrlVmCfg_IsShareUserHomeDir function 638
PrlVmCfg_IsSmartGuardEnabled 639
PrlVmCfg_IsSmartGuardEnabled function 639
PrlVmCfg_IsSmartGuardNotifyBeforeCreation 639
PrlVmCfg_IsSmartGuardNotifyBeforeCreation function 639
PrlVmCfg_IsSmartMountDVDsEnabled 640
PrlVmCfg_IsSmartMountDVDsEnabled function 640
PrlVmCfg_IsSmartMountEnabled 641
PrlVmCfg_IsSmartMountEnabled function 641
PrlVmCfg_IsSmartMountNetworkSharesEnabled 641
PrlVmCfg_IsSmartMountNetworkSharesEnabled function 641
PrlVmCfg_IsSmartMountRemovableDrivesEnabled 642
PrlVmCfg_IsSmartMountRemovableDrivesEnabled function 642
PrlVmCfg_IsStartInDetachedWindowEnabled 642
PrlVmCfg_IsStartInDetachedWindowEnabled function 642

PrlVmCfg_IsSwitchOffWindowsLogoEnabled 643
PrlVmCfg_IsSwitchOffWindowsLogoEnabled function 643
PrlVmCfg_IsTemplate 643
PrlVmCfg_IsTemplate function 643
PrlVmCfg_IsTimeSynchronizationEnabled 644
PrlVmCfg_IsTimeSynchronizationEnabled function 644
PrlVmCfg_IsTimeSyncSmartModeEnabled 644
PrlVmCfg_IsTimeSyncSmartModeEnabled function 644
PrlVmCfg_IsToolsAutoUpdateEnabled 645
PrlVmCfg_IsToolsAutoUpdateEnabled function 645
PrlVmCfg_IsUseDefaultAnswers 646
PrlVmCfg_IsUseDefaultAnswers function 646
PrlVmCfg_IsUseDesktop 646
PrlVmCfg_IsUseDesktop function 646
PrlVmCfg_IsUseDocuments 647
PrlVmCfg_IsUseDocuments function 647
PrlVmCfg_IsUseDownloads 647
PrlVmCfg_IsUseDownloads function 647
PrlVmCfg_IsUseMovies 648
PrlVmCfg_IsUseMovies function 648
PrlVmCfg_IsUseMusic 648
PrlVmCfg_IsUseMusic function 648
PrlVmCfg_IsUsePictures 649
PrlVmCfg_IsUsePictures function 649
PrlVmCfg_IsUserDefinedSharedFoldersEnabled 649
PrlVmCfg_IsUserDefinedSharedFoldersEnabled function 649
PrlVmCfg_IsVerticalSynchronizationEnabled 505
PrlVmCfg_IsVerticalSynchronizationEnabled function 505
PrlVmCfg_IsVNCEncryptionEnabled 650
PrlVmCfg_IsVNCEncryptionEnabled function 650
PrlVmCfg_Set3DAccelerationMode 650
PrlVmCfg_Set3DAccelerationMode function 650
PrlVmCfg_SetActionOnGuestCrash 651
PrlVmCfg_SetActionOnGuestCrash function 651
PrlVmCfg_SetActionOnStopMode 652
PrlVmCfg_SetActionOnStopMode function 652
PrlVmCfg_SetActionOnWindowClose 652
PrlVmCfg_SetActionOnWindowClose function 652
PrlVmCfg_SetAdaptiveHypervisorEnabled 653
PrlVmCfg_SetAdaptiveHypervisorEnabled function 653
PrlVmCfg_SetAllowSelectBootDevice 653
PrlVmCfg_SetAllowSelectBootDevice function 653
PrlVmCfg_SetAppTemplateList 654
PrlVmCfg_SetAppTemplateList function 654
PrlVmCfg_SetAutoApplyIpOnly 654
PrlVmCfg_SetAutoApplyIpOnly function 654
PrlVmCfg_SetAutoCompressEnabled 655
PrlVmCfg_SetAutoCompressEnabled function 655
PrlVmCfg_SetAutoCompressInterval 656
PrlVmCfg_SetAutoCompressInterval function 656
PrlVmCfg_SetAutoStart 656
PrlVmCfg_SetAutoStart function 656
PrlVmCfg_SetAutoStartDelay 657
PrlVmCfg_SetAutoStartDelay function 657
PrlVmCfg_SetAutoStop 657
PrlVmCfg_SetAutoStop function 657
PrlVmCfg_SetBackgroundPriority 658
PrlVmCfg_SetBackgroundPriority function 658
PrlVmCfg_SetBatteryStatusEnabled 658
PrlVmCfg_SetBatteryStatusEnabled function 658
PrlVmCfg_SetCapabilitiesMask 659
PrlVmCfg_SetCapabilitiesMask function 659
PrlVmCfg_SetCloseAppOnShutdown 659
PrlVmCfg_SetCloseAppOnShutdown function 659
PrlVmCfg_SetConfirmationsList 660
PrlVmCfg_SetConfirmationsList function 660
PrlVmCfg_SetCpuAccelLevel 661
PrlVmCfg_SetCpuAccelLevel function 661
PrlVmCfg_SetCpuCount 661
PrlVmCfg_SetCpuCount function 661
PrlVmCfg_SetCpuHotplugEnabled 662
PrlVmCfg_SetCpuHotplugEnabled function 662
PrlVmCfg_SetCpuLimit 662
PrlVmCfg_SetCpuLimit function 662
PrlVmCfg_SetCpuLimitEx 663
PrlVmCfg_SetCpuLimitEx function 663
PrlVmCfg_SetCpuMask 663
PrlVmCfg_SetCpuMask function 663

PrlVmCfg_SetCpuMode 664
PrlVmCfg_SetCpuMode function 664
PrlVmCfg_SetCpuUnits 664
PrlVmCfg_SetCpuUnits function 664
PrlVmCfg_SetCustomProperty 665
PrlVmCfg_SetCustomProperty function 665
PrlVmCfg_SetDefaultConfig 666
PrlVmCfg_SetDefaultConfig function 666
PrlVmCfg_SetDescription 667
PrlVmCfg_SetDescription function 667
PrlVmCfg_SetDisableAPICSign 667
PrlVmCfg_SetDisableAPICSign function 667
PrlVmCfg_SetDisableSpeakerSign 668
PrlVmCfg_SetDisableSpeakerSign function 668
PrlVmCfg_SetDiskCacheWriteBack 668
PrlVmCfg_SetDiskCacheWriteBack function 668
PrlVmCfg_SetDnsServers 669
PrlVmCfg_SetDnsServers function 669
PrlVmCfg_SetEfiEnabled 669
PrlVmCfg_SetEfiEnabled function 669
PrlVmCfg_SetExternalBootDevice 670
PrlVmCfg_SetExternalBootDevice function 670
PrlVmCfg_SetFeaturesMask 671
PrlVmCfg_SetFeaturesMask function 671
PrlVmCfg_SetForegroundPriority 671
PrlVmCfg_SetForegroundPriority function 671
PrlVmCfg_SetFreeDiskSpaceRatio 672
PrlVmCfg_SetFreeDiskSpaceRatio function 672
PrlVmCfg_SetGuestSharingAutoMount 672
PrlVmCfg_SetGuestSharingAutoMount function 672
PrlVmCfg_SetGuestSharingEnabled 673
PrlVmCfg_SetGuestSharingEnabled function 673
PrlVmCfg_SetGuestSharingEnableSpotlight 674
PrlVmCfg_SetGuestSharingEnableSpotlight function 674
PrlVmCfg_SetHighAvailabilityEnabled 674
PrlVmCfg_SetHighAvailabilityEnabled function 674
PrlVmCfg_SetHighAvailabilityPriority 675
PrlVmCfg_SetHighAvailabilityPriority function 675
PrlVmCfg_SetHighResolutionEnabled 506
PrlVmCfg_SetHighResolutionEnabled function 506
PrlVmCfg_SetHostname 676
PrlVmCfg_SetHostname function 676
PrlVmCfg_SetHostSharingEnabled 676
PrlVmCfg_SetHostSharingEnabled function 676
PrlVmCfg_SetIcon 677
PrlVmCfg_SetIcon function 677
PrlVmCfg_SetIoLimit 677
PrlVmCfg_SetIoLimit function 677
PrlVmCfg_SetIoPriority 678
PrlVmCfg_SetIoPriority function 678
PrlVmCfg_SetIopsLimit 678
PrlVmCfg_SetIopsLimit function 678
PrlVmCfg_SetIsolatedVmEnabled 679
PrlVmCfg_SetIsolatedVmEnabled function 679
PrlVmCfg_SetLockGuestOnSuspendEnabled 679
PrlVmCfg_SetLockGuestOnSuspendEnabled function 679
PrlVmCfg_SetLockInFullScreenMode 680
PrlVmCfg_SetLockInFullScreenMode function 680
PrlVmCfg_SetLongerBatteryLifeEnabled 681
PrlVmCfg_SetLongerBatteryLifeEnabled function 681
PrlVmCfg_SetMapSharedFoldersOnLetters 681
PrlVmCfg_SetMapSharedFoldersOnLetters function 681
PrlVmCfg_SetMemGuaranteeSize 682
PrlVmCfg_SetMemGuaranteeSize function 682
PrlVmCfg_SetName 682
PrlVmCfg_SetName function 682
PrlVmCfg_SetNestedVirtualizationEnabled 683
PrlVmCfg_SetNestedVirtualizationEnabled function 683
PrlVmCfg_SetNetfilterMode 683
PrlVmCfg_SetNetfilterMode function 683
PrlVmCfg_SetNetworkRateList 684
PrlVmCfg_SetNetworkRateList function 684
PrlVmCfg_SetNodeMask 685
PrlVmCfg_SetNodeMask function 685
PrlVmCfg_SetOfflineManagementEnabled 685
PrlVmCfg_SetOfflineManagementEnabled function 685
PrlVmCfg_SetOfflineServices 686
PrlVmCfg_SetOfflineServices function 686

PrlVmCfg_SetOptimizeModifiersMode 686
PrlVmCfg_SetOptimizeModifiersMode function 686
PrlVmCfg_SetOptionsOnGuestCrash 687
PrlVmCfg_SetOptionsOnGuestCrash function 687
PrlVmCfg_SetOsResInFullScrMode 687
PrlVmCfg_SetOsResInFullScrMode function 687
PrlVmCfg_SetOsTemplate 688
PrlVmCfg_SetOsTemplate function 688
PrlVmCfg_SetOsVersion 688
PrlVmCfg_SetOsVersion function 688
PrlVmCfg_SetPMUVirtualizationEnabled 689
PrlVmCfg_SetPMUVirtualizationEnabled function 689
PrlVmCfg_SetProfile 690
PrlVmCfg_SetProfile function 690
PrlVmCfg_SetRamHotplugEnabled 690
PrlVmCfg_SetRamHotplugEnabled function 690
PrlVmCfg_SetRamSize 691
PrlVmCfg_SetRamSize function 691
PrlVmCfg_SetRateBound 691
PrlVmCfg_SetRateBound function 691
PrlVmCfg_SetResource 692
PrlVmCfg_SetResource function 692
PrlVmCfg_SetScrResEnabled 692
PrlVmCfg_SetScrResEnabled function 692
PrlVmCfg_SetSearchDomains 693
PrlVmCfg_SetSearchDomains function 693
PrlVmCfg_SetShareAllHostDisks 694
PrlVmCfg_SetShareAllHostDisks function 694
PrlVmCfg_SetShareClipboard 694
PrlVmCfg_SetShareClipboard function 694
PrlVmCfg_SetSharedProfileEnabled 695
PrlVmCfg_SetSharedProfileEnabled function 695
PrlVmCfg_SetShareUserHomeDir 695
PrlVmCfg_SetShareUserHomeDir function 695
PrlVmCfg_SetSmartGuardEnabled 696
PrlVmCfg_SetSmartGuardEnabled function 696
PrlVmCfg_SetSmartGuardInterval 697
PrlVmCfg_SetSmartGuardInterval function 697
PrlVmCfg_SetSmartGuardMaxSnapshotsCount 697
PrlVmCfg_SetSmartGuardMaxSnapshotsCount function 697
PrlVmCfg_SetSmartGuardNotifyBeforeCreation 698
PrlVmCfg_SetSmartGuardNotifyBeforeCreation function 698
PrlVmCfg_SetSmartMountDVDsEnabled 699
PrlVmCfg_SetSmartMountDVDsEnabled function 699
PrlVmCfg_SetSmartMountEnabled 699
PrlVmCfg_SetSmartMountEnabled function 699
PrlVmCfg_SetSmartMountNetworkSharesEnabled 700
PrlVmCfg_SetSmartMountNetworkSharesEnabled function 700
PrlVmCfg_SetSmartMountRemovableDrivesEnabled 700
PrlVmCfg_SetSmartMountRemovableDrivesEnabled function 700
PrlVmCfg_SetStartInDetachedWindowEnabled 701
PrlVmCfg_SetStartInDetachedWindowEnabled function 701
PrlVmCfg_SetStartLoginMode 701
PrlVmCfg_SetStartLoginMode function 701
PrlVmCfg_SetStartUserCreds 702
PrlVmCfg_SetStartUserCreds function 702
PrlVmCfg_SetSwitchOffWindowsLogoEnabled 702
PrlVmCfg_SetSwitchOffWindowsLogoEnabled function 702
PrlVmCfg_SetSystemFlags 703
PrlVmCfg_SetSystemFlags function 703
PrlVmCfg_SetTemplateSign 704
PrlVmCfg_SetTemplateSign function 704
PrlVmCfg_SetTimeSynchronizationEnabled 704
PrlVmCfg_SetTimeSynchronizationEnabled function 704
PrlVmCfg_SetTimeSyncInterval 705
PrlVmCfg_SetTimeSyncInterval function 705
PrlVmCfg_SetTimeSyncSmartModeEnabled 706
PrlVmCfg_SetTimeSyncSmartModeEnabled function 706
PrlVmCfg_SetToolsAutoUpdateEnabled 706
PrlVmCfg_SetToolsAutoUpdateEnabled function 706
PrlVmCfg_SetUndoDisksMode 707
PrlVmCfg_SetUndoDisksMode function 707
PrlVmCfg_SetUseDefaultAnswers 707
PrlVmCfg_SetUseDefaultAnswers function 707
PrlVmCfg_SetUseDesktop 708
PrlVmCfg_SetUseDesktop function 708
PrlVmCfg_SetUseDocuments 709
PrlVmCfg_SetUseDocuments function 709

PrlVmCfg_SetUseDownloads 709
PrlVmCfg_SetUseDownloads function 709
PrlVmCfg_SetUseMovies 710
PrlVmCfg_SetUseMovies function 710
PrlVmCfg_SetUseMusic 710
PrlVmCfg_SetUseMusic function 710
PrlVmCfg_SetUsePictures 711
PrlVmCfg_SetUsePictures function 711
PrlVmCfg_SetUserDefinedSharedFoldersEnabled 711
PrlVmCfg_SetUserDefinedSharedFoldersEnabled function 711
PrlVmCfg_SetUuid 712
PrlVmCfg_SetUuid function 712
PrlVmCfg_SetVerticalSynchronizationEnabled 506
PrlVmCfg_SetVerticalSynchronizationEnabled function 506
PrlVmCfg_SetVideoRamSize 712
PrlVmCfg_SetVideoRamSize function 712
PrlVmCfg_SetVmType 713
PrlVmCfg_SetVmType function 713
PrlVmCfg_SetVNCHostName 713
PrlVmCfg_SetVNCHostName function 713
PrlVmCfg_SetVNCMode 714
PrlVmCfg_SetVNCMode function 714
PrlVmCfg_SetVNCPassword 714
PrlVmCfg_SetVNCPassword function 714
PrlVmCfg_SetVNCPort 715
PrlVmCfg_SetVNCPort function 715
PrlVmCfg_SetWindowMode 715
PrlVmCfg_SetWindowMode function 715
PrlVmDev_Connect 754
PrlVmDev_Connect function 754
PrlVmDev_CopyImage 754
PrlVmDev_CopyImage function 754
PrlVmDev_Create 755
PrlVmDev_Create function 755
PrlVmDev_CreateImage 755
PrlVmDev_CreateImage function 755
PrlVmDev_Disconnect 756
PrlVmDev_Disconnect function 756
PrlVmDev_GetDescription 757
PrlVmDev_GetDescription function 757
PrlVmDev_GetEmulatedType 757
PrlVmDev_GetEmulatedType function 757
PrlVmDev_GetFriendlyName 758
PrlVmDev_GetFriendlyName function 758
PrlVmDev_GetIfaceType 759
PrlVmDev_GetIfaceType function 759
PrlVmDev_GetImagePath 760
PrlVmDev_GetImagePath function 760
PrlVmDev_GetIndex 760
PrlVmDev_GetIndex function 760
PrlVmDev_GetOutputFile 761
PrlVmDev_GetOutputFile function 761
PrlVmDev_GetStackIndex 761
PrlVmDev_GetStackIndex function 761
PrlVmDev_GetSubType 762
PrlVmDev_GetSubType function 762
PrlVmDev_GetSysName 763
PrlVmDev_GetSysName function 763
PrlVmDev_GetType 763
PrlVmDev_GetType function 763
PrlVmDev_IsConnected 764
PrlVmDev_IsConnected function 764
PrlVmDev_IsEnabled 764
PrlVmDev_IsEnabled function 764
PrlVmDev_IsPassthrough 765
PrlVmDev_IsPassthrough function 765
PrlVmDev_IsRemote 765
PrlVmDev_IsRemote function 765
PrlVmDev_Remove 766
PrlVmDev_Remove function 766
PrlVmDev_ResizeImage 766
PrlVmDev_ResizeImage function 766
PrlVmDev_SetConnected 767
PrlVmDev_SetConnected function 767
PrlVmDev_SetDefaultStackIndex 768
PrlVmDev_SetDefaultStackIndex function 768
PrlVmDev_SetDescription 768
PrlVmDev_SetDescription function 768

PrlVmDev_SetEmulatedType 769
PrlVmDev_SetEmulatedType function 769
PrlVmDev_SetEnabled 770
PrlVmDev_SetEnabled function 770
PrlVmDev_SetFriendlyName 770
PrlVmDev_SetFriendlyName function 770
PrlVmDev_SetIfaceType 771
PrlVmDev_SetIfaceType function 771
PrlVmDev_SetImagePath 771
PrlVmDev_SetImagePath function 771
PrlVmDev_SetIndex 772
PrlVmDev_SetIndex function 772
PrlVmDev_SetOutputFile 772
PrlVmDev_SetOutputFile function 772
PrlVmDev_SetPassthrough 773
PrlVmDev_SetPassthrough function 773
PrlVmDev_SetRemote 773
PrlVmDev_SetRemote function 773
PrlVmDev_SetStackIndex 774
PrlVmDev_SetStackIndex function 774
PrlVmDev_SetSubType 774
PrlVmDev_SetSubType function 774
PrlVmDev_SetSysName 775
PrlVmDev_SetSysName function 775
PrlVmDevHd_AddPartition 382
PrlVmDevHd_AddPartition function 382
PrlVmDevHd_GetDiskSize 383
PrlVmDevHd_GetDiskSize function 383
PrlVmDevHd_GetDiskType 384
PrlVmDevHd_GetDiskType function 384
PrlVmDevHd_GetEncryption 384
PrlVmDevHd_GetEncryption function 384
PrlVmDevHd_GetMountPoint 385
PrlVmDevHd_GetMountPoint function 385
PrlVmDevHd_GetPartition 385
PrlVmDevHd_GetPartition function 385
PrlVmDevHd_GetPartitionsCount 386
PrlVmDevHd_GetPartitionsCount function 386
PrlVmDevHd_GetSerialNumber 387
PrlVmDevHd_GetSerialNumber function 387
PrlVmDevHd_GetSizeOnDisk 387
PrlVmDevHd_GetSizeOnDisk function 387
PrlVmDevHd_GetStorageURL 388
PrlVmDevHd_GetStorageURL function 388
PrlVmDevHd_IsAutoCompressEnabled 389
PrlVmDevHd_IsAutoCompressEnabled function 389
PrlVmDevHd_IsSplitted 389
PrlVmDevHd_IsSplitted function 389
PrlVmDevHd_SetAutoCompressEnabled 390
PrlVmDevHd_SetAutoCompressEnabled function 390
PrlVmDevHd_SetDiskSize 391
PrlVmDevHd_SetDiskSize function 391
PrlVmDevHd_SetDiskType 391
PrlVmDevHd_SetDiskType function 391
PrlVmDevHd_SetEncryption 392
PrlVmDevHd_SetEncryption function 392
PrlVmDevHd_SetMountPoint 392
PrlVmDevHd_SetMountPoint function 392
PrlVmDevHd_SetSerialNumber 393
PrlVmDevHd_SetSerialNumber function 393
PrlVmDevHd_SetSplitted 394
PrlVmDevHd_SetSplitted function 394
PrlVmDevHd_SetStorageURL 394
PrlVmDevHd_SetStorageURL function 394
PrlVmDevHdEncryption_GetKeyId 441
PrlVmDevHdEncryption_GetKeyId function 441
PrlVmDevHdEncryption_SetKeyId 441
PrlVmDevHdEncryption_SetKeyId function 441
PrlVmDevHdPart_GetSysName 397
PrlVmDevHdPart_GetSysName function 397
PrlVmDevHdPart_Remove 398
PrlVmDevHdPart_Remove function 398
PrlVmDevHdPart_SetSysName 398
PrlVmDevHdPart_SetSysName function 398
PrlVmDevNet_GenerateMacAddr 401
PrlVmDevNet_GenerateMacAddr function 401
PrlVmDevNet_GetAdapterType 401
PrlVmDevNet_GetAdapterType function 401

PrlVmDevNet_GetBoundAdapterIndex 402
PrlVmDevNet_GetBoundAdapterIndex function 402
PrlVmDevNet_GetBoundAdapterName 402
PrlVmDevNet_GetBoundAdapterName function 402
PrlVmDevNet_GetDefaultGateway 403
PrlVmDevNet_GetDefaultGateway function 403
PrlVmDevNet_GetDefaultGatewayIPv6 404
PrlVmDevNet_GetDefaultGatewayIPv6 function 404
PrlVmDevNet_GetDnsServers 405
PrlVmDevNet_GetDnsServers function 405
PrlVmDevNet_GetFirewallDefaultPolicy 405
PrlVmDevNet_GetFirewallDefaultPolicy function 405
PrlVmDevNet_GetFirewallRuleList 406
PrlVmDevNet_GetFirewallRuleList function 406
PrlVmDevNet_GetHostInterfaceName 395
PrlVmDevNet_GetHostInterfaceName function 395
PrlVmDevNet_GetMacAddress 406
PrlVmDevNet_GetMacAddress function 406
PrlVmDevNet_GetMacAddressCanonical 396
PrlVmDevNet_GetMacAddressCanonical function 396
PrlVmDevNet_GetNetAddresses 407
PrlVmDevNet_GetNetAddresses function 407
PrlVmDevNet_GetSearchDomains 408
PrlVmDevNet_GetSearchDomains function 408
PrlVmDevNet_GetVirtualNetworkId 408
PrlVmDevNet_GetVirtualNetworkId function 408
PrlVmDevNet_IsAutoApply 409
PrlVmDevNet_IsAutoApply function 409
PrlVmDevNet_IsConfigureWithDhcp 410
PrlVmDevNet_IsConfigureWithDhcp function 410
PrlVmDevNet_IsConfigureWithDhcpIPv6 410
PrlVmDevNet_IsConfigureWithDhcpIPv6 function 410
PrlVmDevNet_IsFirewallEnabled 411
PrlVmDevNet_IsFirewallEnabled function 411
PrlVmDevNet_IsPktFilterPreventIpSpoof 411
PrlVmDevNet_IsPktFilterPreventIpSpoof function 411
PrlVmDevNet_IsPktFilterPreventMacSpoof 412
PrlVmDevNet_IsPktFilterPreventMacSpoof function 412
PrlVmDevNet_SetAdapterType 412
PrlVmDevNet_SetAdapterType function 412
PrlVmDevNet_SetAutoApply 413
PrlVmDevNet_SetAutoApply function 413
PrlVmDevNet_SetBoundAdapterIndex 414
PrlVmDevNet_SetBoundAdapterIndex function 414
PrlVmDevNet_SetBoundAdapterName 414
PrlVmDevNet_SetBoundAdapterName function 414
PrlVmDevNet_SetConfigureWithDhcp 415
PrlVmDevNet_SetConfigureWithDhcp function 415
PrlVmDevNet_SetConfigureWithDhcpIPv6 416
PrlVmDevNet_SetConfigureWithDhcpIPv6 function 416
PrlVmDevNet_SetDefaultGateway 416
PrlVmDevNet_SetDefaultGateway function 416
PrlVmDevNet_SetDefaultGatewayIPv6 417
PrlVmDevNet_SetDefaultGatewayIPv6 function 417
PrlVmDevNet_SetDnsServers 418
PrlVmDevNet_SetDnsServers function 418
PrlVmDevNet_SetFirewallDefaultPolicy 419
PrlVmDevNet_SetFirewallDefaultPolicy function 419
PrlVmDevNet_SetFirewallEnabled 419
PrlVmDevNet_SetFirewallEnabled function 419
PrlVmDevNet_SetFirewallRuleList 420
PrlVmDevNet_SetFirewallRuleList function 420
PrlVmDevNet_SetHostInterfaceName 396
PrlVmDevNet_SetHostInterfaceName function 396
PrlVmDevNet_SetMacAddress 420
PrlVmDevNet_SetMacAddress function 420
PrlVmDevNet_SetNetAddresses 421
PrlVmDevNet_SetNetAddresses function 421
PrlVmDevNet_SetPktFilterPreventIpSpoof 422
PrlVmDevNet_SetPktFilterPreventIpSpoof function 422
PrlVmDevNet_SetPktFilterPreventMacSpoof 422
PrlVmDevNet_SetPktFilterPreventMacSpoof function 422
PrlVmDevNet_SetPktFilterPreventPromisc 423
PrlVmDevNet_SetPktFilterPreventPromisc function 423
PrlVmDevNet_SetSearchDomains 423
PrlVmDevNet_SetSearchDomains function 423
PrlVmDevNet_SetVirtualNetworkId 424
PrlVmDevNet_SetVirtualNetworkId function 424

PrlVmDevSerial_GetSocketMode 426
PrlVmDevSerial_GetSocketMode function 426
PrlVmDevSerial_SetSocketMode 427
PrlVmDevSerial_SetSocketMode function 427
PrlVmDevSound_GetMixerDev 428
PrlVmDevSound_GetMixerDev function 428
PrlVmDevSound_GetOutputDev 428
PrlVmDevSound_GetOutputDev function 428
PrlVmDevSound_SetMixerDev 429
PrlVmDevSound_SetMixerDev function 429
PrlVmDevSound_SetOutputDev 430
PrlVmDevSound_SetOutputDev function 430
PrlVmDevUsb_GetAutoconnectOption 431
PrlVmDevUsb_GetAutoconnectOption function 431
PrlVmDevUsb_SetAutoconnectOption 431
PrlVmDevUsb_SetAutoconnectOption function 431
PrlVmGuest_GetNetworkSettings 716
PrlVmGuest_GetNetworkSettings function 716
PrlVmGuest_Logout 717
PrlVmGuest_Logout function 717
PrlVmGuest_RunProgram 718
PrlVmGuest_RunProgram function 718
PrlVmInfo_GetAccessRights 720
PrlVmInfo_GetAccessRights function 720
PrlVmInfo_GetAdditionState 720
PrlVmInfo_GetAdditionState function 720
PrlVmInfo_GetState 721
PrlVmInfo_GetState function 721
PrlVmInfo_IsInvalid 721
PrlVmInfo_IsInvalid function 721
PrlVmInfo_IsInvalidEx 722
PrlVmInfo_IsInvalidEx function 722
PrlVmInfo_IsVmWaitingForAnswer 723
PrlVmInfo_IsVmWaitingForAnswer function 723
PrlVmInfo_IsVncServerStarted 723
PrlVmInfo_IsVncServerStarted function 723
PrlVmToolsInfo_GetState 724
PrlVmToolsInfo_GetState function 724
PrlVmToolsInfo_GetVersion 725
PrlVmToolsInfo_GetVersion function 725
PRN_LPT_DEVICE enumeration member 1016
PRN_UNKNOWN_DEVICE enumeration member 1016
PRN_USB_DEVICE enumeration member 1016
PRNVM_ALLOW_TO_AUTO_DECREASE_HDD_SIZE enumeration member 1022
PRNVM_CREATE_FROM_LION_RECOVERY_PARTITION enumeration member 1022
PRNVM_PRESERVE_DISK enumeration member 1022
PRPM_RUN_PROGRAM_AND_RETURN_IMMEDIATELY enumeration member 930
PRPM_RUN_PROGRAM_ENTER enumeration member 930
PRPM_RUN_PROGRAM_IN_SHELL enumeration member 930
PRR_BACKUP enumeration member 1019
PRR_BOOT_CAMP enumeration member 1019
PRR_CONNECTION enumeration member 1019
PRR_HANG_LOCKUP enumeration member 1019
PRR_KEYBOARD enumeration member 1019
PRR_LAST enumeration member 1019
PRR_MIGRATION enumeration member 1019
PRR_NETWORK enumeration member 1019
PRR_OTHER enumeration member 1019
PRR_PARALLELS_TOOLS enumeration member 1019
PRR_PERFORMANCE enumeration member 1019
PRR_PRODUCT_REGISTRATION enumeration member 1019
PRR_SNAPSHOTS enumeration member 1019
PRR_SOUND enumeration member 1019
PRR_SUPPORT_REQUEST enumeration member 1019
PRR_UNKNOWN enumeration member 1019
PRR_USB enumeration member 1019
PRR_VIDEO_3D_GRAPHICS enumeration member 1019
PRS_NEW_PACKED enumeration member 1019
PRS_OLD_XML_BASED enumeration member 1019
PRT_AUTOMATIC_DETECTED_REPORT enumeration member 1020
PRT_AUTOMATIC_DISPATCHER_GENERATED_REPORT enumeration member 1020
PRT_AUTOMATIC_GUEST_GENERATED_REPORT enumeration member 1020
PRT_AUTOMATIC_INSTALLED_SOFTWARE_REPORT enumeration member 1020
PRT_AUTOMATIC_STATISTICS_REPORT enumeration

member 1020

PRT_AUTOMATIC_TRANSPORTER_REPORT enumeration member 1020

PRT_AUTOMATIC_UTW7AGENT_POST_MIGRATION enumeration member 1020

PRT_AUTOMATIC_UTW7AGENT_PRE_MIGRATION enumeration member 1020

PRT_AUTOMATIC_VM_GENERATED_REPORT enumeration member 1020

PRT_AUTOMATIC_VZ_STATISTICS_REPORT enumeration member 1020

PRT_LAST enumeration member 1020

PRT_REPORT_TYPE_UNDEFINED enumeration member 1020

PRT_USER_DEFINED_ON_CONNECTED_SERVER enumeration member 1020

PRT_USER_DEFINED_ON_CONTAINER_REPORT enumeration member 1020

PRT_USER_DEFINED_ON_DISCONNECTED_SERVER enumeration member 1020

PRT_USER_DEFINED_ON_MOBILE_FEEDBACK_REPORT enumeration member 1020

PRT_USER_DEFINED_ON_MOBILE_REPORT enumeration member 1020

PRT_USER_DEFINED_ON_NOT_RESPONDING_VM_REPORT enumeration member 1020

PRT_USER_DEFINED_ON_RUNNING_VM_REPORT enumeration member 1020

PRT_USER_DEFINED_ON_STOPPED_VM_REPORT enumeration member 1020

PRT_USER_DEFINED_TRANSPORTER_REPORT enumeration member 1020

PRVF_IGNORE_HA_CLUSTER enumeration member 1022

PRVF_KEEP_OTHERS_PERMISSIONS enumeration member 1022

PRVF_REGENERATE_SRC_VM_UUID enumeration member 1022

PRVF_REGENERATE_VM_UUID enumeration member 1022

PSCT_AUTO enumeration member 1025

PSCT_STEREO enumeration member 1025

PSCT_SURROUND_5_1 enumeration member 1025

PSE_DIRECTORY enumeration member 992

PSE_DRIVE enumeration member 992

PSE_FILE enumeration member 992

PSF_FORCE enumeration member 935

PSF_NOFORCE enumeration member 935

PSHF_DONT_WAIT_TO_COMPLETE enumeration member 1025

PSHF_FORCE_SHUTDOWN enumeration member 1025

PSHF_SUSPEND_VM_TO_PRAM enumeration member 1025

PSIA_OPEN_DEFAULT enumeration member 1027

PSIA_OPEN_IN_GUEST enumeration member 1027

PSIA_OPEN_IN_HOST enumeration member 1027

PSL_HIGH_SECURITY enumeration member 1024

PSL_LOW_SECURITY enumeration member 1024

PSL_NORMAL_SECURITY enumeration member 1024

PSM_ACPI enumeration member 936

PSM_KILL enumeration member 936

PSM_LAST enumeration member 936

PSM_SHUTDOWN enumeration member 936

PSM_VM_SAFE_START enumeration member 935

PSM_VM_START enumeration member 935

PSM_VM_START_FOR_COMPACT enumeration member 935

PSM_VM_START_LAST_ITEM enumeration member 935

PSP_SERIAL_SOCKET_CLIENT enumeration member 1024

PSP_SERIAL_SOCKET_SERVER enumeration member 1024

PSPF_PASSWD_CRYPTED enumeration member 1025

PSS_NOT_INSTALLED enumeration member 1024

PSS_STARTED enumeration member 1024

PSS_STOPPED enumeration member 1024

PSS_UNKNOWN enumeration member 1024

PSSF_SKIP_RESUME enumeration member 1026

PST_VM_HIBERNATE enumeration member 995

PST_VM_REBOOT enumeration member 995

PST_VM_SHUTDOWN enumeration member 995

PST_VM_SUSPEND enumeration member 995

PSV_CALC_BOOT_CAMP_SIZE enumeration member 936

PSVM_AUTO enumeration member 1023

PSVM_IGNORE_ASPECT_RATIO enumeration member 1023

PSVM_KEEP_ASPECT_RATIO enumeration member 1023

PSVM_KEEP_ASPECT_RATIO_BY_EXPANDING enumeration member 1023

PSVM_OFF enumeration member 1023

PTBS_DEFAULT enumeration member 1026

PTBS_HIDE enumeration member 1026

PTBS_SHOW enumeration member 1026
PTBS_UNKNOWN enumeration member 1026
PTIS_RECORD_ACTIVE enumeration member 1027
PTIS_RECORD_CANCELED enumeration member 1027
PTIS_RECORD_CLEARED enumeration member 1027
PTIS_RECORD_DATA enumeration member 1027
PTIS_RECORD_EMPTY enumeration member 1027
PTIS_RECORD_FLAGS enumeration member 1027
PTIS_RECORD_INFO enumeration member 1027
PTIS_RECORD_NAME enumeration member 1027
PTIS_RECORD_OWNER enumeration member 1027
PTIS_RECORD_REMOVED enumeration member 1027
PTIS_RECORD_STATE enumeration member 1027
PTIS_RECORD_TEXT enumeration member 1027
PTIS_RECORD_TIME enumeration member 1027
PTIS_RECORD_UID enumeration member 1027
PTS_ABSOLUTE_MOUSE enumeration member 932
PTS_INSTALLED enumeration member 936
PTS_MOVING_CURSOR enumeration member 932
PTS_NO_CURSOR enumeration member 932
PTS_NOT_INSTALLED enumeration member 936
PTS_OUTDATED enumeration member 936
PTS_POSSIBLY_INSTALLED enumeration member 936
PTS_RELATIVE_MOUSE enumeration member 932
PTS_SLIDING_CURSOR enumeration member 932
PTS_SLIDING_MOUSE enumeration member 932
PTS_UNKNOWN enumeration member 936
PTU_CMD_INVALID enumeration member 1028
PTU_CMD_PTU_ALIVE enumeration member 1028
PTU_FLG_ZERO enumeration member 1028
PUD_ASK_USER_WHAT_TODO enumeration member 1029
PUD_COMMIT_CHANGES enumeration member 1029
PUD_CONNECT_TO_GUEST_OS enumeration member 1029
PUD_CONNECT_TO_PRIMARY_OS enumeration member 1029
PUD_CONNECTED_AUTOMATICALLY enumeration member 1030
PUD_CONNECTED_MANUALLY enumeration member 1030
PUD_DISABLE_UNDO_DISKS enumeration member 1029
PUD_PROMPT_BEHAVIOUR enumeration member 1029
PUD_REVERSE_CHANGES enumeration member 1029
PUDT_APPLE_IETH enumeration member 1030
PUDT_APPLE_IPAD enumeration member 1030
PUDT_APPLE_IPHONE enumeration member 1030
PUDT_APPLE_IPOD enumeration member 1030
PUDT_ATAPI_STORAGE enumeration member 1030
PUDT_AUDIO enumeration member 1030
PUDT_BLUETOOTH enumeration member 1030
PUDT_COMMUNICATION enumeration member 1030
PUDT_DISK_STORAGE enumeration member 1030
PUDT_FOTO enumeration member 1030
PUDT_GARMIN_GPS enumeration member 1030
PUDT_HUB enumeration member 1030
PUDT_KEYBOARD enumeration member 1030
PUDT_MOUSE enumeration member 1030
PUDT_OTHER enumeration member 1030
PUDT_PRINTER enumeration member 1030
PUDT_RIM_BLACKBERRY enumeration member 1030
PUDT_SCANNER enumeration member 1030
PUDT_SMART_CARD enumeration member 1030
PUDT_VIDEO enumeration member 1030
PUDT_WIRELESS enumeration member 1030
PUPLF_ACTIVATE_PERMANENT_LICENSE_OFFLINE enumeration member 1029
PUPLF_ACTIVATE_PERMANENT_LICENSE_ONLINE enumeration member 1029
PUPLF_ACTIVATE_PRECACHED_KEY enumeration member 1029
PUPLF_CHECK_KEY_ONLY enumeration member 1029
PUPLF_CHECK_KEY_ONLY_VOLUME enumeration member 1029
PUPLF_DEACTIVATE_CURRENT_LICENSE enumeration member 1029
PUPLF_DEACTIVATE_CURRENT_LICENSE_OFFLINE_ALLOWED enumeration member 1029
PUPLF_KICK_TO_UDPATE_CURR_FILE_LICENSE enumeration member 1029
PUPLF_REGISTER_CURRENT_LICENSE enumeration member 1029
PUPLF_REMOVE_PRECACHED_KEY enumeration member 1029
PUPLF_STORE_PRECACHED_KEY enumeration member

1029
PVA_ACCELERATION_DISABLED enumeration member 923
PVA_ACCELERATION_HIGH enumeration member 923
PVA_ACCELERATION_NORMAL enumeration member 923
PVA_POSTSTART enumeration member 924
PVA_POSTSTOP enumeration member 924
PVA_PRESTART enumeration member 924
PVA_PRESTOP enumeration member 924
PVBT_CT_PLOOP enumeration member 926
PVBT_CT_VZFS enumeration member 926
PVBT_CT_VZWIN enumeration member 926
PVBT_VM enumeration member 926
PVC_ALL enumeration member 928
PVC_BOOT_OPTION enumeration member 928
PVC_CD_DVD_ROM enumeration member 928
PVC_COLOR_BLUE enumeration member 927
PVC_COLOR_GREEN enumeration member 927
PVC_COLOR_GREY enumeration member 927
PVC_COLOR_ORANGE enumeration member 927
PVC_COLOR_PURPLE enumeration member 927
PVC_COLOR_RED enumeration member 927
PVC_COLOR_YELLOW enumeration member 927
PVC_CPU enumeration member 928
PVC_FLOPPY_DISK enumeration member 928
PVC_GENERAL_PARAMETERS enumeration member 928
PVC_GENERIC_PCI enumeration member 928
PVC_HARD_DISK enumeration member 928
PVC_IDE_DEVICES enumeration member 928
PVC_LAST_SECTION enumeration member 928
PVC_LICENSE_RESTRICTIONS enumeration member 928
PVC_MAIN_MEMORY enumeration member 928
PVC_NETWORK_ADAPTER enumeration member 928
PVC_NO_COLOR enumeration member 927
PVC_OFFLINE_MANAGEMENT_SETTINGS enumeration member 928
PVC_PARALLEL_PORT enumeration member 928
PVC_REMOTE_DISPLAY enumeration member 928
PVC_SATA_DEVICES enumeration member 928
PVC_SCSI_DEVICES enumeration member 928
PVC_SERIAL_PORT enumeration member 928
PVC_SHARED_FOLDERS enumeration member 928
PVC_SOUND enumeration member 928
PVC_VALIDATE_CHANGES_ONLY enumeration member 928
PVC_VIDEO_MEMORY enumeration member 928
PVCF_DESTROY_HDD_BUNDLE enumeration member 927
PVCF_DESTROY_HDD_BUNDLE_FORCE enumeration member 927
PVCF_DETACH_HDD_BUNDLE enumeration member 927
PVCF_RENAME_EXT_DISKS enumeration member 927
PVCF_WAIT_FOR_APPLY enumeration member 927
PVL_FIREWIRE_DRIVE enumeration member 930
PVL_LOCAL_FS enumeration member 930
PVL_REMOTE_FS enumeration member 930
PVL_UNKNOWN enumeration member 930
PVL_USB_DRIVE enumeration member 930
PVM_DONT_RESUME_VM enumeration member 931
PVM_GET_SECURITY_LEVEL 1671
PVM_GET_SECURITY_LEVEL macro 1671
PVMSF_GUEST_OS_INFORMATION_ONLY enumeration member 935
PVMSF_HOST_DISK_SPACE_USAGE_ONLY enumeration member 935
PVMSL_HIGH_SECURITY enumeration member 931
PVMSL_LOW_SECURITY enumeration member 931
PVMSL_NORMAL_SECURITY enumeration member 931
PVMT_CHANGE_SID enumeration member 931
PVMT_CLONE_MODE enumeration member 931
PVMT_COLD_MIGRATION enumeration member 931
PVMT_DIRECT_DATA_CONNECTION enumeration member 931
PVMT_DONT_CREATE_DISK enumeration member 931
PVMT_DONT_RESUME_VM enumeration member 931
PVMT_HOT_MIGRATION enumeration member 931
PVMT_IGNORE_EXISTING_BUNDLE enumeration member 931
PVMT_REMOVE_SOURCE_BUNDLE enumeration member 931
PVMT_SEND_DISK_MAP enumeration member 931
PVMT_SWITCH_TEMPLATE enumeration member 931
PVMT_UNCOMPRESSED enumeration member 931
PVMT_WARM_MIGRATION enumeration member 931

PVN_BRIDGED_ETHERNET enumeration member 1013
PVN_HOST_ONLY enumeration member 1013
PVR_PRIORITY_HIGH enumeration member 933
PVR_PRIORITY_LOW enumeration member 933
PVR_PRIORITY_NORMAL enumeration member 933
PVRCS_CONNECTED enumeration member 1100
PVRCS_CONNECTING enumeration member 1100
PVRCS_DISCONNECTED enumeration member 1100
PVRCS_NO_CONNECTION enumeration member 1100
PVS_GET_GUEST_NUMBER 1671
PVS_GET_GUEST_NUMBER macro 1671
PVS_GET_GUEST_TYPE 1671
PVS_GET_GUEST_TYPE macro 1671
PVS_GUEST Macros 1576
PVS_GUEST_ARCH_I386 1581
PVS_GUEST_ARCH_I386 macro 1581
PVS_GUEST_ARCH_OTHER 1581
PVS_GUEST_ARCH_OTHER macro 1581
PVS_GUEST_ARCH_X86_64 1582
PVS_GUEST_ARCH_X86_64 macro 1582
PVS_GUEST_TYPE_ANDROID 1582
PVS_GUEST_TYPE_ANDROID macro 1582
PVS_GUEST_TYPE_CHROMEOS 1582
PVS_GUEST_TYPE_CHROMEOS macro 1582
PVS_GUEST_TYPE_FREEBSD 1582
PVS_GUEST_TYPE_FREEBSD macro 1582
PVS_GUEST_TYPE_LINUX 1582
PVS_GUEST_TYPE_LINUX macro 1582
PVS_GUEST_TYPE_MACOS 1583
PVS_GUEST_TYPE_MACOS macro 1583
PVS_GUEST_TYPE_MSDOS 1583
PVS_GUEST_TYPE_MSDOS macro 1583
PVS_GUEST_TYPE_NAME_ANDROID 1583
PVS_GUEST_TYPE_NAME_ANDROID macro 1583
PVS_GUEST_TYPE_NAME_CHROMEOS 1583
PVS_GUEST_TYPE_NAME_CHROMEOS macro 1583
PVS_GUEST_TYPE_NAME_FREEBSD 1583
PVS_GUEST_TYPE_NAME_FREEBSD macro 1583
PVS_GUEST_TYPE_NAME_LINUX 1584
PVS_GUEST_TYPE_NAME_LINUX macro 1584
PVS_GUEST_TYPE_NAME_MACOS 1584
PVS_GUEST_TYPE_NAME_MACOS macro 1584
PVS_GUEST_TYPE_NAME_MSDOS 1584
PVS_GUEST_TYPE_NAME_MSDOS macro 1584
PVS_GUEST_TYPE_NAME_NETWARE 1584
PVS_GUEST_TYPE_NAME_NETWARE macro 1584
PVS_GUEST_TYPE_NAME_OS2 1585
PVS_GUEST_TYPE_NAME_OS2 macro 1585
PVS_GUEST_TYPE_NAME_OTHER 1585
PVS_GUEST_TYPE_NAME_OTHER macro 1585
PVS_GUEST_TYPE_NAME_SOLARIS 1585
PVS_GUEST_TYPE_NAME_SOLARIS macro 1585
PVS_GUEST_TYPE_NAME_WINDOWS 1585
PVS_GUEST_TYPE_NAME_WINDOWS macro 1585
PVS_GUEST_TYPE_NETWARE 1585
PVS_GUEST_TYPE_NETWARE macro 1585
PVS_GUEST_TYPE_OS2 1586
PVS_GUEST_TYPE_OS2 macro 1586
PVS_GUEST_TYPE_OTHER 1586
PVS_GUEST_TYPE_OTHER macro 1586
PVS_GUEST_TYPE_SOLARIS 1586
PVS_GUEST_TYPE_SOLARIS macro 1586
PVS_GUEST_TYPE_WINDOWS 1586
PVS_GUEST_TYPE_WINDOWS macro 1586
PVS_GUEST_VER_ANDROID_2_x 1586
PVS_GUEST_VER_ANDROID_2_x macro 1586
PVS_GUEST_VER_ANDROID_3_x 1587
PVS_GUEST_VER_ANDROID_3_x macro 1587
PVS_GUEST_VER_ANDROID_4_0 1587
PVS_GUEST_VER_ANDROID_4_0 macro 1587
PVS_GUEST_VER_ANDROID_LAST 1587
PVS_GUEST_VER_ANDROID_LAST macro 1587
PVS_GUEST_VER_ANDROID_OTHER 1587
PVS_GUEST_VER_ANDROID_OTHER macro 1587
PVS_GUEST_VER_BSD_4X 1587
PVS_GUEST_VER_BSD_4X macro 1587
PVS_GUEST_VER_BSD_5X 1588
PVS_GUEST_VER_BSD_5X macro 1588

PVS_GUEST_VER_BSD_6X 1588
PVS_GUEST_VER_BSD_6X macro 1588
PVS_GUEST_VER_BSD_7X 1588
PVS_GUEST_VER_BSD_7X macro 1588
PVS_GUEST_VER_BSD_8X 1588
PVS_GUEST_VER_BSD_8X macro 1588
PVS_GUEST_VER_BSD_LAST 1588
PVS_GUEST_VER_BSD_LAST macro 1588
PVS_GUEST_VER_BSD_OTHER 1589
PVS_GUEST_VER_BSD_OTHER macro 1589
PVS_GUEST_VER_CHROMEOS_1x 1589
PVS_GUEST_VER_CHROMEOS_1x macro 1589
PVS_GUEST_VER_CHROMEOS_LAST 1589
PVS_GUEST_VER_CHROMEOS_LAST macro 1589
PVS_GUEST_VER_CHROMEOS_OTHER 1589
PVS_GUEST_VER_CHROMEOS_OTHER macro 1589
PVS_GUEST_VER_DOS_LAST 1590
PVS_GUEST_VER_DOS_LAST macro 1590
PVS_GUEST_VER_DOS_MS622 1590
PVS_GUEST_VER_DOS_MS622 macro 1590
PVS_GUEST_VER_DOS_OTHER 1590
PVS_GUEST_VER_DOS_OTHER macro 1590
PVS_GUEST_VER_LIN_CENTOS 1590
PVS_GUEST_VER_LIN_CENTOS macro 1590
PVS_GUEST_VER_LIN_CENTOS_7 1590
PVS_GUEST_VER_LIN_CENTOS_7 macro 1590
PVS_GUEST_VER_LIN_CLOUDLINUX 1591
PVS_GUEST_VER_LIN_CLOUDLINUX macro 1591
PVS_GUEST_VER_LIN_CLOUDLINUX_7 1591
PVS_GUEST_VER_LIN_CLOUDLINUX_7 macro 1591
PVS_GUEST_VER_LIN_DEBIAN 1591
PVS_GUEST_VER_LIN_DEBIAN macro 1591
PVS_GUEST_VER_LIN_FEDORA 1591
PVS_GUEST_VER_LIN_FEDORA macro 1591
PVS_GUEST_VER_LIN_FEDORA_5 1592
PVS_GUEST_VER_LIN_FEDORA_5 macro 1592
PVS_GUEST_VER_LIN_KRNL_24 1592
PVS_GUEST_VER_LIN_KRNL_24 macro 1592
PVS_GUEST_VER_LIN_KRNL_26 1592
PVS_GUEST_VER_LIN_KRNL_26 macro 1592
PVS_GUEST_VER_LIN_LAST 1592
PVS_GUEST_VER_LIN_LAST macro 1592
PVS_GUEST_VER_LIN_MAGEIA 1592
PVS_GUEST_VER_LIN_MAGEIA macro 1592
PVS_GUEST_VER_LIN_MANDRAKE 1593
PVS_GUEST_VER_LIN_MANDRAKE macro 1593
PVS_GUEST_VER_LIN_MINT 1593
PVS_GUEST_VER_LIN_MINT macro 1593
PVS_GUEST_VER_LIN_OPENSUSE 1593
PVS_GUEST_VER_LIN_OPENSUSE macro 1593
PVS_GUEST_VER_LIN_OTHER 1593
PVS_GUEST_VER_LIN_OTHER macro 1593
PVS_GUEST_VER_LIN_PSBM 1594
PVS_GUEST_VER_LIN_PSBM macro 1594
PVS_GUEST_VER_LIN_REDHAT 1594
PVS_GUEST_VER_LIN_REDHAT macro 1594
PVS_GUEST_VER_LIN_REDHAT_7 1594
PVS_GUEST_VER_LIN_REDHAT_7 macro 1594
PVS_GUEST_VER_LIN_RH_LEGACY 1594
PVS_GUEST_VER_LIN_RH_LEGACY macro 1594
PVS_GUEST_VER_LIN_RHLES3 1594
PVS_GUEST_VER_LIN_RHLES3 macro 1594
PVS_GUEST_VER_LIN_SLES11 1595
PVS_GUEST_VER_LIN_SLES11 macro 1595
PVS_GUEST_VER_LIN_SLES12 1595
PVS_GUEST_VER_LIN_SLES12 macro 1595
PVS_GUEST_VER_LIN_SLES9 1595
PVS_GUEST_VER_LIN_SLES9 macro 1595
PVS_GUEST_VER_LIN_SUSE 1595
PVS_GUEST_VER_LIN_SUSE macro 1595
PVS_GUEST_VER_LIN_UBUNTU 1595
PVS_GUEST_VER_LIN_UBUNTU macro 1595
PVS_GUEST_VER_LIN_VZLINUX 1596
PVS_GUEST_VER_LIN_VZLINUX macro 1596
PVS_GUEST_VER_LIN_VZLINUX_7 1596
PVS_GUEST_VER_LIN_VZLINUX_7 macro 1596
PVS_GUEST_VER_LIN_XANDROS 1596
PVS_GUEST_VER_LIN_XANDROS macro 1596

PVS_GUEST_VER_MACOS_LAST 1596
PVS_GUEST_VER_MACOS_LAST macro 1596
PVS_GUEST_VER_MACOS_LEOPARD 1596
PVS_GUEST_VER_MACOS_LEOPARD macro 1596
PVS_GUEST_VER_MACOS_SNOW_LEOPARD 1597
PVS_GUEST_VER_MACOS_SNOW_LEOPARD macro 1597
PVS_GUEST_VER_MACOS_TIGER 1597
PVS_GUEST_VER_MACOS_TIGER macro 1597
PVS_GUEST_VER_MACOS_UNIVERSAL 1597
PVS_GUEST_VER_MACOS_UNIVERSAL macro 1597
PVS_GUEST_VER_NAME_ANDROID_4_0 1597
PVS_GUEST_VER_NAME_ANDROID_4_0 macro 1597
PVS_GUEST_VER_NAME_ANDROID_OTHER 1598
PVS_GUEST_VER_NAME_ANDROID_OTHER macro 1598
PVS_GUEST_VER_NAME_BSD_4X 1598
PVS_GUEST_VER_NAME_BSD_4X macro 1598
PVS_GUEST_VER_NAME_BSD_5X 1598
PVS_GUEST_VER_NAME_BSD_5X macro 1598
PVS_GUEST_VER_NAME_BSD_6X 1598
PVS_GUEST_VER_NAME_BSD_6X macro 1598
PVS_GUEST_VER_NAME_BSD_7X 1598
PVS_GUEST_VER_NAME_BSD_7X macro 1598
PVS_GUEST_VER_NAME_BSD_8X 1599
PVS_GUEST_VER_NAME_BSD_8X macro 1599
PVS_GUEST_VER_NAME_BSD_OTHER 1599
PVS_GUEST_VER_NAME_BSD_OTHER macro 1599
PVS_GUEST_VER_NAME_CHROMEOS_1x 1599
PVS_GUEST_VER_NAME_CHROMEOS_1x macro 1599
PVS_GUEST_VER_NAME_CHROMEOS_OTHER 1599
PVS_GUEST_VER_NAME_CHROMEOS_OTHER macro 1599
PVS_GUEST_VER_NAME_DOS_MS622 1599
PVS_GUEST_VER_NAME_DOS_MS622 macro 1599
PVS_GUEST_VER_NAME_DOS_OTHER 1600
PVS_GUEST_VER_NAME_DOS_OTHER macro 1600
PVS_GUEST_VER_NAME_LIN_CENTOS 1600
PVS_GUEST_VER_NAME_LIN_CENTOS macro 1600
PVS_GUEST_VER_NAME_LIN_CLOUDLINUX 1600
PVS_GUEST_VER_NAME_LIN_CLOUDLINUX macro 1600
PVS_GUEST_VER_NAME_LIN_DEBIAN 1600
PVS_GUEST_VER_NAME_LIN_DEBIAN macro 1600
PVS_GUEST_VER_NAME_LIN_FEDORA 1600
PVS_GUEST_VER_NAME_LIN_FEDORA macro 1600
PVS_GUEST_VER_NAME_LIN_FEDORA_5 1601
PVS_GUEST_VER_NAME_LIN_FEDORA_5 macro 1601
PVS_GUEST_VER_NAME_LIN_KRNL_24 1601
PVS_GUEST_VER_NAME_LIN_KRNL_24 macro 1601
PVS_GUEST_VER_NAME_LIN_KRNL_26 1601
PVS_GUEST_VER_NAME_LIN_KRNL_26 macro 1601
PVS_GUEST_VER_NAME_LIN_MAGEIA 1601
PVS_GUEST_VER_NAME_LIN_MAGEIA macro 1601
PVS_GUEST_VER_NAME_LIN_MANDRAKE 1601
PVS_GUEST_VER_NAME_LIN_MANDRAKE macro 1601
PVS_GUEST_VER_NAME_LIN_MINT 1602
PVS_GUEST_VER_NAME_LIN_MINT macro 1602
PVS_GUEST_VER_NAME_LIN_OPENSUSE 1602
PVS_GUEST_VER_NAME_LIN_OPENSUSE macro 1602
PVS_GUEST_VER_NAME_LIN_OTHER 1602
PVS_GUEST_VER_NAME_LIN_OTHER macro 1602
PVS_GUEST_VER_NAME_LIN_PSBM 1602
PVS_GUEST_VER_NAME_LIN_PSBM macro 1602
PVS_GUEST_VER_NAME_LIN_REDHAT 1602
PVS_GUEST_VER_NAME_LIN_REDHAT macro 1602
PVS_GUEST_VER_NAME_LIN_RH_LEGACY 1603
PVS_GUEST_VER_NAME_LIN_RH_LEGACY macro 1603
PVS_GUEST_VER_NAME_LIN_RHLES3 1603
PVS_GUEST_VER_NAME_LIN_RHLES3 macro 1603
PVS_GUEST_VER_NAME_LIN_SLES9 1603
PVS_GUEST_VER_NAME_LIN_SLES9 macro 1603
PVS_GUEST_VER_NAME_LIN_SUSE 1603
PVS_GUEST_VER_NAME_LIN_SUSE macro 1603
PVS_GUEST_VER_NAME_LIN_UBUNTU 1603
PVS_GUEST_VER_NAME_LIN_UBUNTU macro 1603
PVS_GUEST_VER_NAME_LIN_VZLINUX 1604
PVS_GUEST_VER_NAME_LIN_VZLINUX macro 1604
PVS_GUEST_VER_NAME_LIN_XANDROS 1604
PVS_GUEST_VER_NAME_LIN_XANDROS macro 1604
PVS_GUEST_VER_NAME_MACOS_LEOPARD 1604
PVS_GUEST_VER_NAME_MACOS_LEOPARD macro 1604

PVS_GUEST_VER_NAME_MACOS_SNOW_LEOPARD 1604
PVS_GUEST_VER_NAME_MACOS_SNOW_LEOPARD macro 1604
PVS_GUEST_VER_NAME_MACOS_TIGER 1604
PVS_GUEST_VER_NAME_MACOS_TIGER macro 1604
PVS_GUEST_VER_NAME_NET_4X 1605
PVS_GUEST_VER_NAME_NET_4X macro 1605
PVS_GUEST_VER_NAME_NET_5X 1605
PVS_GUEST_VER_NAME_NET_5X macro 1605
PVS_GUEST_VER_NAME_NET_6X 1605
PVS_GUEST_VER_NAME_NET_6X macro 1605
PVS_GUEST_VER_NAME_NET_OTHER 1605
PVS_GUEST_VER_NAME_NET_OTHER macro 1605
PVS_GUEST_VER_NAME_OS2_ECS11 1606
PVS_GUEST_VER_NAME_OS2_ECS11 macro 1606
PVS_GUEST_VER_NAME_OS2_ECS12 1606
PVS_GUEST_VER_NAME_OS2_ECS12 macro 1606
PVS_GUEST_VER_NAME_OS2_OTHER 1606
PVS_GUEST_VER_NAME_OS2_OTHER macro 1606
PVS_GUEST_VER_NAME_OS2_WARP3 1606
PVS_GUEST_VER_NAME_OS2_WARP3 macro 1606
PVS_GUEST_VER_NAME_OS2_WARP4 1606
PVS_GUEST_VER_NAME_OS2_WARP4 macro 1606
PVS_GUEST_VER_NAME_OS2_WARP45 1607
PVS_GUEST_VER_NAME_OS2_WARP45 macro 1607
PVS_GUEST_VER_NAME_OTH_OPENSTEP 1607
PVS_GUEST_VER_NAME_OTH_OPENSTEP macro 1607
PVS_GUEST_VER_NAME_OTH_OTHER 1607
PVS_GUEST_VER_NAME_OTH_OTHER macro 1607
PVS_GUEST_VER_NAME_OTH_QNX 1607
PVS_GUEST_VER_NAME_OTH_QNX macro 1607
PVS_GUEST_VER_NAME_SOL_10 1607
PVS_GUEST_VER_NAME_SOL_10 macro 1607
PVS_GUEST_VER_NAME_SOL_11 1608
PVS_GUEST_VER_NAME_SOL_11 macro 1608
PVS_GUEST_VER_NAME_SOL_9 1608
PVS_GUEST_VER_NAME_SOL_9 macro 1608
PVS_GUEST_VER_NAME_SOL_OPEN 1608
PVS_GUEST_VER_NAME_SOL_OPEN macro 1608
PVS_GUEST_VER_NAME_SOL_OTHER 1608
PVS_GUEST_VER_NAME_SOL_OTHER macro 1608
PVS_GUEST_VER_NAME_WIN_2003 1608
PVS_GUEST_VER_NAME_WIN_2003 macro 1608
PVS_GUEST_VER_NAME_WIN_2008 1609
PVS_GUEST_VER_NAME_WIN_2008 macro 1609
PVS_GUEST_VER_NAME_WIN_2012 1609
PVS_GUEST_VER_NAME_WIN_2012 macro 1609
PVS_GUEST_VER_NAME_WIN_2K 1609
PVS_GUEST_VER_NAME_WIN_2K macro 1609
PVS_GUEST_VER_NAME_WIN_311 1609
PVS_GUEST_VER_NAME_WIN_311 macro 1609
PVS_GUEST_VER_NAME_WIN_95 1609
PVS_GUEST_VER_NAME_WIN_95 macro 1609
PVS_GUEST_VER_NAME_WIN_98 1610
PVS_GUEST_VER_NAME_WIN_98 macro 1610
PVS_GUEST_VER_NAME_WIN_ME 1610
PVS_GUEST_VER_NAME_WIN_ME macro 1610
PVS_GUEST_VER_NAME_WIN_NT 1610
PVS_GUEST_VER_NAME_WIN_NT macro 1610
PVS_GUEST_VER_NAME_WIN_OTHER 1610
PVS_GUEST_VER_NAME_WIN_OTHER macro 1610
PVS_GUEST_VER_NAME_WIN_VISTA 1610
PVS_GUEST_VER_NAME_WIN_VISTA macro 1610
PVS_GUEST_VER_NAME_WIN_WINDOWS7 1611
PVS_GUEST_VER_NAME_WIN_WINDOWS7 macro 1611
PVS_GUEST_VER_NAME_WIN_WINDOWS8 1611
PVS_GUEST_VER_NAME_WIN_WINDOWS8 macro 1611
PVS_GUEST_VER_NAME_WIN_WINDOWS8_1 1611
PVS_GUEST_VER_NAME_WIN_WINDOWS8_1 macro 1611
PVS_GUEST_VER_NAME_WIN_XP 1611
PVS_GUEST_VER_NAME_WIN_XP macro 1611
PVS_GUEST_VER_NET_4X 1611
PVS_GUEST_VER_NET_4X macro 1611
PVS_GUEST_VER_NET_5X 1612
PVS_GUEST_VER_NET_5X macro 1612
PVS_GUEST_VER_NET_6X 1612
PVS_GUEST_VER_NET_6X macro 1612
PVS_GUEST_VER_NET_LAST 1612
PVS_GUEST_VER_NET_LAST macro 1612

PVS_GUEST_VER_NET_OTHER 1612
PVS_GUEST_VER_NET_OTHER macro 1612
PVS_GUEST_VER_OS2_ECS11 1612
PVS_GUEST_VER_OS2_ECS11 macro 1612
PVS_GUEST_VER_OS2_ECS12 1613
PVS_GUEST_VER_OS2_ECS12 macro 1613
PVS_GUEST_VER_OS2_LAST 1613
PVS_GUEST_VER_OS2_LAST macro 1613
PVS_GUEST_VER_OS2_OTHER 1613
PVS_GUEST_VER_OS2_OTHER macro 1613
PVS_GUEST_VER_OS2_WARP3 1613
PVS_GUEST_VER_OS2_WARP3 macro 1613
PVS_GUEST_VER_OS2_WARP4 1614
PVS_GUEST_VER_OS2_WARP4 macro 1614
PVS_GUEST_VER_OS2_WARP45 1614
PVS_GUEST_VER_OS2_WARP45 macro 1614
PVS_GUEST_VER_OTH_LAST 1614
PVS_GUEST_VER_OTH_LAST macro 1614
PVS_GUEST_VER_OTH_OPENSTEP 1614
PVS_GUEST_VER_OTH_OPENSTEP macro 1614
PVS_GUEST_VER_OTH_OTHER 1614
PVS_GUEST_VER_OTH_OTHER macro 1614
PVS_GUEST_VER_OTH_QNX 1615
PVS_GUEST_VER_OTH_QNX macro 1615
PVS_GUEST_VER_SOL_10 1615
PVS_GUEST_VER_SOL_10 macro 1615
PVS_GUEST_VER_SOL_11 1615
PVS_GUEST_VER_SOL_11 macro 1615
PVS_GUEST_VER_SOL_9 1615
PVS_GUEST_VER_SOL_9 macro 1615
PVS_GUEST_VER_SOL_LAST 1615
PVS_GUEST_VER_SOL_LAST macro 1615
PVS_GUEST_VER_SOL_OPEN 1616
PVS_GUEST_VER_SOL_OPEN macro 1616
PVS_GUEST_VER_SOL_OTHER 1616
PVS_GUEST_VER_SOL_OTHER macro 1616
PVS_GUEST_VER_WIN_2003 1616
PVS_GUEST_VER_WIN_2003 macro 1616
PVS_GUEST_VER_WIN_2008 1616
PVS_GUEST_VER_WIN_2008 macro 1616
PVS_GUEST_VER_WIN_2012 1617
PVS_GUEST_VER_WIN_2012 macro 1617
PVS_GUEST_VER_WIN_2016 1617
PVS_GUEST_VER_WIN_2016 macro 1617
PVS_GUEST_VER_WIN_2K 1617
PVS_GUEST_VER_WIN_2K macro 1617
PVS_GUEST_VER_WIN_311 1617
PVS_GUEST_VER_WIN_311 macro 1617
PVS_GUEST_VER_WIN_95 1617
PVS_GUEST_VER_WIN_95 macro 1617
PVS_GUEST_VER_WIN_98 1618
PVS_GUEST_VER_WIN_98 macro 1618
PVS_GUEST_VER_WIN_LAST 1618
PVS_GUEST_VER_WIN_LAST macro 1618
PVS_GUEST_VER_WIN_ME 1618
PVS_GUEST_VER_WIN_ME macro 1618
PVS_GUEST_VER_WIN_NT 1618
PVS_GUEST_VER_WIN_NT macro 1618
PVS_GUEST_VER_WIN_OTHER 1618
PVS_GUEST_VER_WIN_OTHER macro 1618
PVS_GUEST_VER_WIN_VISTA 1619
PVS_GUEST_VER_WIN_VISTA macro 1619
PVS_GUEST_VER_WIN_WINDOWS10 1619
PVS_GUEST_VER_WIN_WINDOWS10 macro 1619
PVS_GUEST_VER_WIN_WINDOWS7 1619
PVS_GUEST_VER_WIN_WINDOWS7 macro 1619
PVS_GUEST_VER_WIN_WINDOWS8 1619
PVS_GUEST_VER_WIN_WINDOWS8 macro 1619
PVS_GUEST_VER_WIN_WINDOWS8_1 1620
PVS_GUEST_VER_WIN_WINDOWS8_1 macro 1620
PVS_GUEST_VER_WIN_XP 1620
PVS_GUEST_VER_WIN_XP macro 1620
PVS_MAKE_GUEST_VER 1672
PVS_MAKE_GUEST_VER macro 1672
PVSC_ACPI_ERROR enumeration member 1034
PVSC_DMAR_ERROR enumeration member 1034
PVSC_DMAR_NOT_FOUND enumeration member 1034
PVSC_ENABLED enumeration member 1034

PVSC_HARDWARE_ERROR enumeration member 1034
PVSC_INTERNAL_ERROR enumeration member 1034
PVSC_NOT_PRESENT enumeration member 1034
PVSS_CUSTOM enumeration member 934
PVSS_OPTIMIZED_FOR_TIME_MACHINE enumeration member 934
PVT_CT enumeration member 937
PVT_VM enumeration member 937
PVTF_CT enumeration member 937
PVTF_VM enumeration member 937
PWC_BOTTOM_LEFT_CORNER enumeration member 1035
PWC_BOTTOM_RIGHT_CORNER enumeration member 1035
PWC_TOP_LEFT_CORNER enumeration member 1035
PWC_TOP_RIGHT_CORNER enumeration member 1035
PWC_VM_ASK_USER enumeration member 924
PWC_VM_DO_NOTHING enumeration member 924
PWC_VM_PAUSE enumeration member 924
PWC_VM_SHUTDOWN enumeration member 924
PWC_VM_STOP enumeration member 924
PWC_VM_SUSPEND enumeration member 924
PWC_VM_UNKNOWN_ACTION enumeration member 924
PWM_DEFAULT_WINDOW_MODE enumeration member 937
PWM_FULL_SCREEN_WINDOW_MODE enumeration member 937
PWM_WINDOWED_WINDOW_MODE enumeration member 937

R

REINSTALL_RESET_PWDB enumeration member 980
REINSTALL_SKIP_BACKUP enumeration member 980
REINSTALL_SKIP_SCRIPTS enumeration member 980
Remote Desktop Access 777
RTT_RUNNING_TASK_CLONE_VM enumeration member 1023
RTT_RUNNING_TASK_COPY_IMAGE enumeration member 1023
RTT_RUNNING_TASK_CREATE_IMAGE enumeration member 1023
RTT_RUNNING_TASK_CREATE_VM enumeration member 1023
RTT_RUNNING_TASK_DELETE_VM enumeration member 1023
RTT_RUNNING_TASK_MIGRATE_VM enumeration member 1023
RTT_RUNNING_TASK_REGISTER_VM enumeration member 1023
RTT_RUNNING_TASK_RESTORE_VM enumeration member 1023
RTT_RUNNING_TASK_UNKNOWN enumeration member 1023
RTT_RUNNING_TASK_UNREG_VM enumeration member 1023

S

Secondary Display Capture 778
SET_FS_EXECUTE_DISABLED 1672
SET_FS_EXECUTE_DISABLED macro 1672
SET_FS_EXECUTE_ENABLED 1672
SET_FS_EXECUTE_ENABLED macro 1672
SET_FS_READ_DISABLED 1672
SET_FS_READ_DISABLED macro 1672
SET_FS_READ_ENABLED 1673
SET_FS_READ_ENABLED macro 1673
SET_FS_WRITE_DISABLED 1673
SET_FS_WRITE_DISABLED macro 1673
SET_FS_WRITE_ENABLED 1673
SET_FS_WRITE_ENABLED macro 1673
Structs, Records, Enums 837
System Functions 727

T

TEV_CONNECTED enumeration member 1034
TEV_DATA_RECEIVED enumeration member 1034
TEV_DISCONNECTED enumeration member 1034
TO_PARALLELS_API_VER 1673
TO_PARALLELS_API_VER macro 1673
Types 782

V

VCMMD_COMMAND_RUNTIME enumeration member 1032
Virtual Device Functions 752
VIRTUAL_MACHINE_ADDITION_STATE 1104
VIRTUAL_MACHINE_ADDITION_STATE enumeration 1104

VIRTUAL_MACHINE_ADDITION_STATE_PTR 836
VIRTUAL_MACHINE_ADDITION_STATE_PTR type 836
VIRTUAL_MACHINE_STATE 1104
VIRTUAL_MACHINE_STATE enumeration 1104
VIRTUAL_MACHINE_STATE_PTR 836
VIRTUAL_MACHINE_STATE_PTR type 836
VMAS_BACKUPING enumeration member 1103
VMAS_CLONING enumeration member 1103
VMAS_MAX enumeration member 1103
VMAS_MOVING enumeration member 1103
VMAS_NOSTATE enumeration member 1103
VMAS_RESTORING_FROM_BACKUP enumeration member 1103
VMP_DESIGNER enumeration member 1033
VMP_GAMES enumeration member 1033
VMP_PRODUCTIVITY enumeration member 1033
VMP_SOFTWARE_DEVELOPER enumeration member 1033
VMS_COMPACTING enumeration member 1103
VMS_CONTINUING enumeration member 1103
VMS_DELETING_STATE enumeration member 1103
VMS_MIGRATING enumeration member 1103
VMS_MOUNTED enumeration member 1103
VMS_PAUSED enumeration member 1103
VMS_PAUSING enumeration member 1103
VMS_RECONNECTING enumeration member 1103
VMS_RESETTING enumeration member 1103
VMS_RESTORING enumeration member 1103
VMS_RESUMING enumeration member 1103
VMS_RUNNING enumeration member 1103
VMS_SNAPSHOTING enumeration member 1103
VMS_STARTING enumeration member 1103
VMS_STOPPED enumeration member 1103
VMS_STOPPING enumeration member 1103
VMS_SUSPENDED enumeration member 1103
VMS_SUSPENDING enumeration member 1103
VMS_SUSPENDING_SYNC enumeration member 1103
VMS_UNKNOWN enumeration member 1103

W

WINDOW_ALL_EDGE_MASK enumeration member 1035
WINDOW_BOTTOM_EDGE_MASK enumeration member 1035
WINDOW_LEFT_EDGE_MASK enumeration member 1035
WINDOW_RIGHT_EDGE_MASK enumeration member 1035
WINDOW_TOP_EDGE_MASK enumeration member 1035