Parallels™

Parallels Virtual Automation 4.5 Agent

XML API Reference

Copyright © 1999-2010 Parallels, Inc.

ISBN: N/A
Parallels Holdings, Ltd.
c/o Parallels Software, Inc.
13755 Sunrise Valley Drive
Suite 600
Herndon, VA 20171
USA
Tel: +1 (703) 815 5670
Fax: +1 (703) 815 5675

Copyright © 1999-2010 Parallels Holdings, Ltd. and its affiliates. All rights reserved.
Parallels, Coherence, Parallels Transporter, Parallels Compressor, Parallels Desktop, and Parallels Explorer are registered trademarks of Parallels Software International, Inc. Virtuozzo, Plesk, HSPcomplete, and corresponding logos are trademarks of Parallels Holdings, Ltd. The Parallels logo is a trademark of Parallels Holdings, Ltd.
This product is based on a technology that is the subject matter of a number of patent pending applications.
Virtuozzo is a patented virtualization technology protected by U.S. patents 7,099,948; 7,076,633; 6,961,868 and having patents pending in the U.S.
Plesk and HSPcomplete are patented hosting technologies protected by U.S. patents 7,099,948; 7,076,633 and having patents pending in the U.S.
Distribution of this work or derivative of this work in any form is prohibited unless prior written permission is obtained from the copyright holder.
Apple, Bonjour, Finder, Mac, Macintosh, and Mac OS are trademarks of Apple Inc.
Microsoft, Windows, Microsoft Windows, MS-DOS, Windows NT, Windows 95, Windows 98, Windows 2000, Windows XP, Windows 2003 Server, Windows Vista, Windows 2008, Microsoft SQL Server, Microsoft Desktop Engine (MSDE), and Microsoft Management Console are trademarks or registered trademarks of Microsoft Corporation.
Linux is a registered trademark of Linus Torvalds.
Red Hat is a registered trademark of Red Hat Software, Inc.
SUSE is a registered trademark of Novell, Inc.
Solaris is a registered trademark of Sun Microsystems, Inc.
X Window System is a registered trademark of X Consortium, Inc.
UNIX is a registered trademark of The Open Group.
IBM DB2 is a registered trademark of International Business Machines Corp.
SSH and Secure Shell are trademarks of SSH Communications Security, Inc.
MegaRAID is a registered trademark of American Megatrends, Inc.
PowerEdge is a trademark of Dell Computer Corporation.
eComStation is a trademark of Serenity Systems International.
FreeBSD is a registered trademark of the FreeBSD Foundation.
Intel, Pentium, Celeron, and Intel Core are trademarks or registered trademarks of Intel Corporation.
OS/2 Warp is a registered trademark of International Business Machines Corporation.
VMware is a registered trademark of VMware, Inc.
All other marks and names mentioned herein may be trademarks of their respective owners.

Contents

Preface 5

About This Guide 6
Who Should Read This Guide 6
Organization of This Guide 6
How to Use This Guide 7
Documentation Conventions....8
Typographical Conventions....9
Shell Prompts in Command Examples 9
General Conventions 10
Feedback....10

Reference Format 11

XML Message Specifications....11
XML Code Samples 15

Agent Messages 16

Message Header....16
Message Body....21

Base Types and Interfaces 23

Common Types....23
Primitive Types 23
Simple Types....25
Complex Types....27
Interfaces 64
alertm....64
authm....73
backup_storagem....106
backupm 107
computerm....141
data_storagem....157
devm 160
env_samplem....189
envm 193
event_log 231
filer 235
firewallm 263
licensem....272
mailer....287
networkm....295
op_log....313
packagem....318
perf_mon 352
proc_info 365
processm....374
relocator....378
res_log 401

resourcem 414
scheduler....426
server_group....440
servicem....457
sessionm 468
userm 484
Events 504
Types 504
Elements 506
System Interface and Special Packets....514
system....514
The progress packet....563

Virtuozzo Containers Types and Interfaces 567

Common Types....568
Simple Types....568
Complex Types....568
Interfaces 579
vzadevm....579
vzaenvm....580
vzarelocator 604
vzanetworkm 606
vzapackagem 612
vzaproc_info....612
vzaprocessm 620
vzaup2date....621
vzasupport 627

Parallels Server Types and Interfaces 634

Common Types....634
Complex Types....634
Interfaces 642
vzpenvm 642
vzprelocator....645

Appendix A: Performance Counters 647

Appendix B: States and Transitions 656

Appendix C: Error Codes 658

Index 671

CHAPTER 1

Preface

In This Chapter

About This Guide 6
Who Should Read This Guide.....6
Organization of This Guide 6
How to Use This Guide 7
Documentation Conventions.....8
Feedback 10

About This Guide

This guide is a complete Parallels Agent XML API reference. The XML API consists of interfaces to the Virtuozzo Containers management functions. An interface provides calls (similar to functions or methods in traditional programming languages) that allow to interact with the Virtuozzo Containers software on the server side. Using the XML API, you can build reliable tools for remote management and monitoring of the physical and virtual servers.

Who Should Read This Guide

This guide is intended for the developers who are writing their own Parallels Agent applications using XML API. This guide should also be used by the developers using Parallels Agent SOAP API. The proxy classes that you will generate using Parallels Agent WSDL specifications will have the same basic structure as the interfaces and calls described in this guide. By examining an XML API documentation, you can get a clear understanding of how to use a corresponding method of a proxy class in your SOAP-based application.

Organization of This Guide

This guide is organized into the following chapters:

Chapter 1, Preface. Provides information about this guide.

Chapter 2, Reference Format. Explains how to use the specifications presented in this guide.

Chapter 3, Agent Messages. Provides a description of the Parallels Agent XML message structure.

Chapter 4, Base Types and Interfaces. Describes the base data types and interfaces. The chapter is divided into sections. The Common Types section (p. 23) describes the base data types that are used throughout the API. The Interfaces section (p. 64) describes the base interfaces and the available API calls. Each API call documentation consists of the XML request and response specifications, the description of the parameters, and one or more XML code samples.

Chapter 5, Virtuozzo Containers Types and Interfaces. This chapter is organized similarly to the Base Types and Interfaces chapter but describes the types and interfaces that are specific to the Virtuozzo Containers management only. Some of these interfaces and types are derived from the base interfaces and types. If a call is not overridden in the derived interface, it will be documented in the base interface only. However, the Virtuozzo Containers specifics will still be documented and the appropriate examples will be provided.

Appendix A: Performance Counters. Provides a complete list of the available performance classes and counters which are used for performance monitoring.

Appendix B: States and Transitions. Provides a complete list of the available server states and transitions.

Appendix C: Error Codes. Provides a complete list of the Parallels Agent error codes, grouped by the interface or the category to which they apply.

How to Use This Guide

You don't have to read the entire XML reference guide from cover to cover, but you should read at least the Preface chapter (you are reading it now), the Reference Format chapter, and the Agent Messages chapter. The information provided in these chapters is essential to understanding the rest of the reference material. To get a better understanding of Parallels Agent and to learn how to write your own client programs using the provided API, you should also read the Parallels Agent Programmer's Guide which is a companion book to this one.

Each XML API interface provides calls to perform operations of a particular type. For example, the `vzaenvm` interface (p. 580) allows to manage Virtuozzo Containers, the `perf_mon` interface (p. 352) allows to monitor the performance of a Virtuozzo Container or the Hardware Node, etc. In this respect, the interfaces in the Agent XML API are similar to classes in traditional OOP languages, and the calls are similar to methods. The names of the interfaces are abbreviations based on the name of the functionality that they provide. For example, `vzaenvm` stands for Virtuozzo Environment Management, `perf_mon` stands for Performance Monitoring, etc. To find the specifications for a particular operation, browse the Interfaces sections of the Base Types and Interfaces chapter or the Virtuozzo Containers Types and Interfaces chapter. Find the interface that interests you and read the introductory section which gives a brief description about the functionality that the interface provides. After that, proceed to the Calls subsection which lists the available calls that the interface provides. Select the call that interests you and proceed to the subsection describing it. In the subsection, you will find the request specifications, the response specifications, the description of the call, and an XML code sample. This should provide you with enough information to use the call in your client application to perform the desired operation. You may also search the guide using keywords.

Documentation Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it.

The table below presents the existing formatting conventions:

Formatting Conventions	Type of information	Example
Special Bold	Items you must select, such as menu options, command buttons or items in a list.	Go to the Resources tab.
	Titles of chapters, sections and subsections.	Read the Basic Administration chapter.
Italics	Used to emphasize the importance of a point, to introduce a term or to designate a command line placeholder, which is to be replaced with a real name or value.	These are the so-called *EZ templates*. To destroy a Container, type `vzctl destroy` *ctid*.
`Monospace`	The names of commands, files and directories.	Use `vzctl start` to start a Container.
`Preformatted`	On-screen computer output in your command line sessions; source code in XML, C++, or other programming languages.	`Saves parameters for Container 101`
`Monospace Bold`	What you type as contrasted with on-screen computer output.	**`# rpm -V virtuozzo-release`**
Key+Key	Key combinations for which the user should press and hold down one key and then press another.	Ctrl+P, Alt+F4

Besides the formatting conventions, you should also know about the document organization convention applied to Parallels documents: chapters in all guides are divided into sections, which, in turn, are subdivided into subsections. For example, About This Guide is a section, and Documentation Conventions is a subsection.

Typographical Conventions

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information	Example
Triangular Bullet(➢)	Step-by-step procedures. You can follow the instructions below to complete a specific task.	*To create a Container:*
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the Resources tab.
	Titles of chapters, sections, and subsections.	Read the Basic Administration chapter.
Italics	Used to emphasize the importance of a point, to introduce a term or to designate a command line placeholder, which is to be replaced with a real name or value.	These are the so-called *EZ templates*. To destroy a Conainer, type `vzctl destroy` *`ctid`*.
`Monospace`	The names of commands, files, and directories.	Use `vzctl start` to start a Container.
`Preformatted`	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	`Saved parameters for Container 101`
`Monospace Bold`	What you type, contrasted with on-screen computer output.	**`# rpm –V virtuozzo-release`**
CAPITALS	Names of keys on the keyboard.	SHIFT, CTRL, ALT
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another.	CTRL+P, ALT+F4

Shell Prompts in Command Examples

Command line examples throughout this guide presume that you are using the Bourne-again shell (bash). Whenever a command can be run as a regular user, we will display it with a dollar sign prompt. When a command is meant to be run as root, we will display it with a hash mark prompt:

Bourne-again shell prompt $

Bourne-again shell root prompt #

General Conventions

Be aware of the following conventions used in this book.

- Chapters in this guide are divided into sections, which, in turn, are subdivided into subsections. For example, Documentation Conventions is a section, and General Conventions is a subsection.
- When following steps or using examples, be sure to type double-quotes ("), left single-quotes (`), and right single-quotes (') exactly as shown.
- The key referred to as RETURN is labeled ENTER on some keyboards.

The root path usually includes the `/bin`, `/sbin`, `/usr/bin` and `/usr/sbin` directories, so the steps in this book show the commands in these directories without absolute path names. Steps that use commands in other, less common, directories show the absolute paths in the examples.

Feedback

If you spot a typo in this guide, or if you have thought of a way to make this guide better, we would love to hear from you!

The ideal place for your comments and suggestions is the Parallels documentation feedback page (http://www.parallels.com/en/support/usersdoc/).

CHAPTER 2

Reference Format

This chapter explains how to use the specifications presented in this guide.

In This Chapter

XML Message Specifications 11
XML Code Samples 15

XML Message Specifications

The XML message specifications in this guide are described using tables, similar to the following example:

Name	Min/Max	Type	Description
`login`			
`{`			
`name`	`0..1`	`base64Binary`	User name.
`domain`	`0..1`	`base64Binary`	Domain.
`realm`	`1..1`	`guid_type`	Realm ID.
`password`	`0..1`	`base64Binary`	User password.
`expiration`	`0..1`	`int`	Custom timeout value.
`}`			

The information in a table is based on a corresponding XML Schema and describes the format of a request or response message, or the format of a data type.

Each row in a table represents an XML element. The elements are displayed in the order they are defined in the XML Schema.

The definitions for the table columns are as follows:

Name. Specifies an XML element name. The curly brackets represent the standard XML Schema `xs:sequence` element. This means that the elements inside the brackets are the child elements of the element that precedes the opening bracket. In our example, the `name`, `domain`, `realm`, `password`, and `expiration` elements are children of the `login` element. The following is a sample XML code, built according to this specification:

```
<login>
  <name>bXluYW1l</name>
  <domain>bXlkb21haW4=</domain>
  <realm>bXlyZWFsbQ==</realm>
  <password>bXlwYXNz</password>
  <expiration>1000</expiration>
</login>
```

In addition, we use square brackets to represent the standard XML Schema `xs:choice` element, as shown in the following example:

Name	Min/Max	Type	Description
`status`			Device status,
`[`			*Denotes a choice between the <up> and the <down> elements.*
`up`	`1..1`	`none`	Enabled.
`down`	`1..1`	`none`	Disabled.
`]`			

This means that the elements inside the brackets are the child elements of the element that precedes the opening bracket but the elements are mutually exclusive -- only one of them can be present in the request.

Min/Max. Specifies the cardinality of an element (the number of its minimum and maximum occurrences) in the following format:

```
minOccurs..maxOccurs
```

- `0` in the first position indicates that the element is optional.
- `1` in the first position indicates that the element is mandatory and must occur at least once.
- A number in the second position indicates the maximum number of occurrences.
- The `[]` (square brackets) in the second position indicate that the number of the element occurrences is unbounded, meaning that the element may occur as many times as necessary in the same XML document at the specified position.

The following examples demonstrate how an element cardinality may be specified:

- `0..1` The element is optional and may occur only once if included in the document.
- `1..1` The element is mandatory. One, and only one, occurrence is expected in the document.
- `0..[]` The element is optional but may occur an unlimited number of times if needed.
- `1..[]` The element is mandatory. At least one occurrence is expected but the element may occur an unlimited number of times if needed.

Type. Specifies the element type. The following element types are used in the schema:

- Standard simple types: `int, string, base64Binary,` etc.
- Custom simple types. These types are usually derived from standard simple types with additional restrictions imposed on them.
- Custom complex types.

Description. The description column contains the element description and provides information about its usage.

Let's now use the schema example from the beginning of this section and build the Agent request message from it. We already built the message body from it earlier. To make it a fully qualified Agent request message, we must also add the interface name to it and the message header. The following example is a complete Agent message that can be sent to the Agent and be processed by it:

```
<?xml version="1.0" encoding="utf-8"?>
<packet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types" version="4.0.0"
id="2">
<data>
<system>
<login xsi:type="ns2:auth_nameType">
<name>QWRtaW5pc3RyYXRvcg==</name>
<realm>00000000-0000-0000-0000-000000000000</realm>
<password>MXEydzNl</password>
</login>
</system>
</data>
</packet>
```

XML Code Samples

Most of the XML API call descriptions have one or more XML code samples. The samples are provided at the end of the section describing a call. You may copy an entire example and paste it into your program to quick-test the call. More than one samples may be provided for calls where different invocation scenarios must be considered. Please note that some values used in the samples may not be suitable for your particular situation and must be substituted with the values appropriate to your setup. The following is a typical XML code sample:

Example:

Retrieving Parallels Agent version number.

Input

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <get_version/>
    </system>
  </data>
</packet>
```

Output

```
<packet id="2c446af2aet18be" time="2006-05-17T09:54:00+0000" priority="0"
version="4.0.0">
  <session>vzl.30100.ba7be334-4804-4494-a9c8-
15149a0438a5.8c446ae612t2cd6</session>
  <target>vzclient4</target>
  <dst>
    <director>gend</director>
  </dst>
  <data>
    <system>
      <version>vzagent-4.0.0</version>
    </system>
  </data>
  <src>
    <director>gend</director>
  </src>
</packet>
```

The Input section contains a complete XML response message built according to the schema definition.

The Output section contains the actual response message received from Agent.

Note: Some of the elements and attributes common to all response messages will be omitted from the Output examples for brevity. These attributes and elements may include `time, priority` from the `<packet>` element, and `<session>`, `<target>`, `<dst>`, `<src>` from the message header, and possibly some other elements and attributes that are not essential to a particular example. The `<data>` element containing the output data will be included in its entirety, unless noted otherwise in the example itself.

Agent Messages

In order to build XML messages correctly and to take full advantage of the available options, it is important to understand the basic building blocks of a message. This section describes how an Agent message is organized, and provides the necessary specifications and examples.

Message Header

The two main sections of any Agent XML message is the *header* and the *body*. The header provides message routing and control information. The body of the message contains the actual request (or response) parameters and data. The `packet` element is the root element of every message. Both the header and the body of a message reside within the same parent `packet` element.

The following table contains the Agent message header specification, as defined in XML Schema.

Message header specification:

Name	Min..Max	Type	Description
`packet`			The root element of an Agent XML message.
`{`			
`cookie`	`0..1`	`string`	User-defined information describing the message, or any other type of information. The data specified here remains unchanged during the request/response operation, i.e. if you put some data into this element in the request message, the response message will contain the same data.

target	0..[]	string	In request messages, this element must contain the name of the operator to which the request should be sent for processing. **Note:** When using the system operator, do not include the target element. The system operator is the only exception. All other operators require this element. The name of the operator is always the same as the name of the corresponding interface that you are using. For example, if you are using a call from the vzaenvm interface, the name of the target operator is also vzaenvm. Multiple targets may be specified if you are including multiple calls in a single request. In response messages, this element contains the name of the client that originated the request (the value is generated and used internally by Agent).
origin	0..1	string	The name of the operator that generated the response. Included in response messages only.
src	0..1	routeType	The source routing information. This field is automatically populated by the director on the server side when a message is routed from the corresponding operator to it. The same information is also duplicated in the dst element (described below) when a response is generated and is sent back to the client.
{			
director	0..1	string	The name of the director to which the target operator belongs.
host	0..1	string	The Agent host ID. Used by Agent to determine the host address. Should be either contained in the Agent configuration (global mapping) or be a result of exclusive connect.
index	0..1	string	For on-demand operators, specifies a particular target.
target	0..1	string	Contains the origin information when a packet is sent remotely.
}			

dst		routeType	The destination routing information. In request messages, use this structure to specify the server to which you want to forward the request. For example, if you are sending a request to the Agent on the Hardware Node but would like the request to be processed inside a Virtuozzo Container, specify the Server ID for the Container using the dst/host parameter. In the response messages, this information is automatically populated by the director on the server side.
{			
director	0..1	string	The name of the director to which the target operator belongs. Populated automatically by the director.
host	0..1	string	Destination Server ID. When using the message forwarding feature, it is used for specifying the ID of the target server.
index	0..1	string	For on-demand operators, specifies a particular target. Populated automatically by the director.
target	0..1	string	Contains the origin information when a packet is sent remotely. Populated automatically by Agent.
}			
session		string	Session ID. In the request messages, this field is used to specify the session that should be used to process the request. In the response messages, the ID indicates the session that was used to process the request. The session ID is obtained from the response message of the system/login (p. 553) API call after a successful login.
}			

The `packet` element may optionally contain attributes described in the following table.

Attributes of the <packet> element:

Attribute	Type	Description
`version`	`string`	Parallels Agent protocol version number. The current protocol version number is 4.0.0. The older 3.0.3 protocol is also supported in Virtuozzo Containers 4.0.
`id`	`string`	Packet ID. If included in a request message, the response will contain the same ID. This allows the response to be correlated with the original request. The attribute must also be included if you want to be notified in case of the request timeout, or if the packet was dropped on the server side for any reason. As a rule of thumb, you should always include this element in all of your outgoing packets. The value should normally be a string containing an integer value, but it can also contain other characters if needed.
`priority`	`string`	Packet priority. Specifies the significance of the message when it is placed into a message queue. The higher the priority value, the less significant the packet is. The value of zero is the default priority. Priorities range from -3000 to 3000. -3000 to -1000 for heavy messages. -999 to 999 for normal messages. 1000 to 3000 for urgent messages.
`time`	`datetime_type`	The time when the packet was sent; in the ISO-8601 format: (e.g. "2007-02-04T08:55:51+0000").
`progress`	`string`	Use this attribute to enable the progress reporting for long operations if you would like to receive intermediate results and to keep track of the request processing. Please note that not all operations actually generate progress reports. The possible values are: `on` -- the progress reporting is on. `off` (default if the attribute is omitted) -- the progress reporting is off. When you turn the progress reporting on, you must also include the `id` attribute (above) specifying the message ID.

log	string	When present, the automatic progress reporting is logged for the operations supporting it. Switch this to “on” if you're planning to start an operation and disconnect from Agent before the operation is completed. By doing so, you'll be able to reconnect later and check the log files for the results of your operation. The requests marked as *Logged Operation* in the XML API Reference support this feature. Possible values are: on -- the logging is turned on. off (default) -- the logging is off.
type	int	*** INTERNAL *** Bit field for the internal type of the message. #define UNFINISHED 0x00000001 #define RESPONSE 0x00000002 #define RESCHEDULE 0x00000004 #define TIMEOUT 0x00000008
timeout	int	The timeout value which will be used for handling this request. The value can be specified in the incoming packet or it can be sent back from the operator, notifying the director about the time it is going to handle it.
timeout_limit	int	*** INTERNAL *** Timeout limit for message processing. Used by an operator in determining the validity of its timeout.
uid	int	*** INTERNAL *** UID of the user sending this packet.

Example:

The following is an example of an Agent message header, built according to the specifications above. In a real message, the values of the XML elements would be substituted with the appropriate names, IDs, etc.

```
<packet version="4.0.0" id="500">
  <cookie>I'm a cookie holding some text</cookie>
  <target>operator_name</target>
  <dst>
    Hosttarget_server_ID</host>
  </dst>
  <session>session_id</session>
</packet>
```

Message Body

Message body contains the actual request or response parameters and data. The `data` element is the root element of the message body tree. It is followed by the name of the interface that you would like to use, the name of the call, and the call parameters.

Note: There must be one and only one `data` element in any given message.

The request message:

The following XML code example is a complete Agent request message. As you already know, the `packet` element is the root element of every Agent message. The `target` element specifies the name of the target operator. The message body begins with the `data` element. The `envm` element specifies the name of the interface. The available interfaces are documented in the Parallels Agent XML API Reference documentation. The `get_info` element is the name of the call. The `config` element specifies that the information about hardware node configuration is requested.

```
<packet version="4.0.0" id="2">
  <target>envm</target>
  <data>
    <envm>
      <get_info>
        <config/>
      </get_info>
    </envm>
  </data>
</packet>
```

The response message:

The following example demonstrates a complete response message. The body of the message begins with the `data` element which is followed by the name of the interface that was used in the corresponding request message, and the return parameters.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/envm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="ac49b7e2c8t6784r580" time="2009-03-11T16:09:33+0000">
<ns1:origin>envm</ns1:origin>
```

```
<ns1:target>vzclient4-b1e6bc1e-4231-b541-819a-191cd7fec5fb</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:envm>
        <ns2:env xsi:type="ns3:envType">
            <ns3:parent_eid>00000000-0000-0000-0000-
000000000000</ns3:parent_eid>
            <ns3:eid>b1e6bc1e-4231-b541-819a-191cd7fec5fb</ns3:eid>
            <ns3:status xsi:type="ns3:env_statusType">
                <ns3:state>6</ns3:state>
            </ns3:status>
            <ns3:alert>0</ns3:alert>
            <ns3:config xsi:type="ns3:env_configType">
                <ns3:name>mccp40.qa.sw.ru</ns3:name>
                <ns3:hostname>mccp40.qa.sw.ru</ns3:hostname>
                <ns3:address>
                    <ns3:ip>10.27.1.174</ns3:ip>
                </ns3:address>
                <ns3:address>
                    <ns3:ip>10.37.130.2</ns3:ip>
                </ns3:address>
                <ns3:address>
                    <ns3:ip>10.37.131.2</ns3:ip>
                </ns3:address>
                <ns3:architecture>x86_64</ns3:architecture>
                <ns3:os xsi:type="ns3:osType">
                    <ns3:platform>Linux</ns3:platform>
                    <ns3:kernel>2.6.18-028stab061.6</ns3:kernel>
                    <ns3:name>Parallels Server Bare Metal 3.0.0</ns3:name>
                </ns3:os>
                <ns3:type>generic</ns3:type>
                <ns3:nameserver>10.27.0.1</ns3:nameserver>
                <ns3:child_type>parallels</ns3:child_type>
                <ns3:child_type>virtuozzo</ns3:child_type>
            </ns3:config>
            <ns3:virtual_config xsi:type="ns3:venv_configType"/>
        </ns2:env>
    </ns2:envm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

The body of a response message may, in general, contain one of the following types of information:

- The actual information requested, as shown in the example above.
- The `<OK/>` element if the call doesn't return any data by definition. The `<OK/>` means that the operation completed successfully.
- An error information, in case of a failure.

A complete XML Schema specification exists for every possible response of every Agent XML API call, and is described in the corresponding section of the Parallels Agent XML API Reference guide.

CHAPTER 3

Base Types and Interfaces

This chapter describes the base XML API types and interfaces. They can be used to perform operations on physical servers (Hardware Nodes),virtual machines, and Virtuozzo Containers. In addition, the Virtuozzo Containers Types and Interfaces chapter (p. 567) contains specifications of the types and interfaces that are specific to the Virtuozzo Containers only; the Parallels Server Types and Interfaces chapter (p. 634) contains specifications of the types and interfaces that are specific to virtual machines only.

In This Chapter

Common Types.....23
Interfaces 64
Events 504
System Interface and Special Packets.....514

Common Types

This chapter describes the common data types. There are three main categories of common types:

- Primitive Types are the most basic data types. These are built-in types, which are defined in the W3C XML Schema Data Types specification.
- Simple Types are custom types that are usually derived from primitive types with additional restrictions imposed on them.
- Complex Types are custom types that can contain attributes and elements.

Each category is described in detail in the following sections.

Primitive Types

Primitive types are the basic building blocks of an XML Schema. The following table describes the most common primitive types used in the Agent XML Schema.

Name	Description
`boolean`	Boolean type. Legal values for boolean are `true`, `false`, `1` (which indicates true), and `0` (which indicates false).
`int`	A signed 32-bit integer.
`long`	A signed 64-bit integer
`double`	A 64-bit floating point.

`string`	A string. Note that unlike char arrays in C, XML strings are NOT null-terminated.
`base64Binary`	Base64-encoded binary data.

Simple Types

Simple types are custom types that can contain a value but cannot contain attributes or elements. Most of the custom simple types have restrictions added to them in order to limit their content. A restriction can limit the type to a specific primitive data type, it can also define a list of enumerated values, or it can define a string pattern that the value must adhere to.

datetime_type

Summary:

Holds a datetime value.

Type specification:

`datetime_type` is derived from `string`

The type complies with ISO-8601, the International Standard for the representation of dates and times. The format is as follows.

`YYYY-MM-DDThh:mm:ssZ`

where:

`YYYY` -- four-digit year.

`MM` -- two-digit month (01 = January, etc.).

`DD` -- two-digit day of month (01 through 31).

The letter `T` in `DDThh` must literally be present to indicate the beginning of the time element.

hh -- two digits of hour (00 through 23, am/pm is NOT allowed).

`mm` -- two digits of minute (00 through 59).

`ss` -- two digits of second (00 through 59).

`Z` -- GMT/UTC offset (+hhmm or -hhmm).

Example:

`2006-05-28T19:05:30-0500` corresponds to May 28, 2006, 19:05:30 (or 7:05:30 pm), US Eastern Standard Time.

eid_type

Summary:

Holds a Server ID value. Server ID is a globally unique identifier that is assigned to every computer (physical or virtual) that has Parallels Agent installed on it. The ID is assigned to a physical server as soon as Virtuozzo Containers software is installed on it. The ID is assigned to a Virtuozzo Container at the time of creation.

Type specification:

Restriction: guid_type (p. 26)

guid_type

Summary:

A globally unique identifier.

Type specification:

`guid_type` is derived from `string`

ip_type

Summary:

IP (Internet Protocol) address expressed as four decimal numbers separated by periods, such as 192.168.1.10

Type specification:

`ip_type` is derived from `string`

privilegeType

Summary:

A security privilege identification.

Type specification:

Restriction: `string`

sidType

Summary:

Security ID.

Type specification:

Restriction: `base64Binary`

transport_type

Summary:

Transport type enumeration.

Type specification:

The enumeration has the following enumerators:

`TCP` -- Transmission Control Protocol.

`UDP` -- User Datagram Protocol.

Complex Types

Complex types are custom types that can contain text, attributes and other elements. This section describes the types that are common to the entire Parallels Agent XML API.

aceType

Summary:

Access control entry.

Type specification:

Name	Min/Max	Type	Description
`type`	`1..1`	`int`	Type of ACE: `0` -- allow `1` -- deny
`sid`	`1..1`	`sidType (p. 27)`	Security identifier of a user or a group.
`rights`	`1..1`	`base64Binary`	Access rights.

Description:

The access control entry (ACE) is an individual record in a DACL (Discretionary Access Control List). It includes the SID (security ID) of a single user or a group along with an access mask that specifies the permissions being granted or denied.

At the time of this writing, you can only set permissions to the entire object (not the individual operations that can be performed on it). This means that the `rights` parameter is not currently used. Simply include an empty `rights` element when setting permissions.

See also:

`security_descriptorType` (p. 56)

alert_dataType

Summary:

Contains alert information.

Type specification:

Extends `event_dataType (p. 35)`

Adds the following elements:

Name	Min/Max	Type	Description
`type`	`1..1`	`int`	`0` -- green alert. `1` -- yellow alert. `2` -- red alert.

Description:

The alert type denotes the severity level. There are four alert levels:

Green alert -- Normal operation. This alert type is normally silent but can still trigger when one of the higher alert levels is canceled and the situation returns to normal.

Yellow alert -- Warning. For a resource allocation alert it means that at least 90% of the specified soft limit was reached.

Red alert -- Critical situation. For a resource alert it means that the current resource usage is above the soft limit and further allocation can be refused at any moment.

The subtypes of `alert_dataType` are used to handle different alert categories, such as resource allocation alerts and cluster-wide alerts.

Subtypes:

`resource_alertType` (p. 64)

`server_group_alertType` (p. 65)

auth_nameType

Summary:

User login information.

Type specification:

Name	Min/Max	Type	Description
`name`	`0..1`	`base64Binary`	The name of a user or a group. When working with LDAP directory entries, the name can be specified as a fully qualified distinguished name or as a plain name (as in "John" for instance). If the name contains a full DN then the entry is assumed to be located in the container specified. If the name is a plain name (as in "John" for instance), the entry is assumed to be located in the default container for users and groups for this realm. To find out what the default DN is, use the `get_realm` call (p. 99).
`domain`	`0..1`	`base64Binary`	Domain name. Currently, this filed is only used to specify the Server ID (eid) when logging into Agent as a user from the Virtuozzo Container Realm (the OS user registry inside the Container).

realm	1..1	guid_type (p. 26)	Realm ID. When adding a user or a group to a realm, specify the target realm ID here.

connection_infoType

Summary:

Contains parameters necessary to connect to a remote computer.

Type specification:

Extends connectivity_infoType **(p. 30)**

Adds the following elements:

Name	Min/Max	Type	Description
login	0..1	auth_nameType (p. 29)	Login information.
password	0..1	base64Binary	Password.

connectivity_infoType

Summary:

Contains the network connectivity information.

Type specification:

Adds the following elements:

Name	Min/Max	Type	Description
protocol	0..1	string	Communication protocol: SSL -- SSL over TCP/IP. TCP -- plain TCP/IP. NamedPipe -- named pipe.
address	1..1	string	IP address.
port	0..1	unsignedInt	Port number.

cpu_loadType

Summary:

Contains CPU load values.

Type specification:

Name	Min/Max	Type	Description
system	1..1	long	CPU used by system processes.
user	1..1	long	CPU used by user processes.
nice	1..1	long	CPU used by "nice" processes.
idle	1..1	long	CPU idle.

cpuType

Summary:

Contains common CPU characteristics.

Type specification:

Name	Min/Max	Type	Description
mhz	1..1	int	CPU Frequency, in MegaHertz.
name	1..1	string	CPU name.
number	1..1	int	Number of CPUs in the system.
cores	1..1	int	Number of cores per CPU.
hyperthreads	1..1	int	Number of hyper-threads per CPU core. The value of 1 indicates that no hyper-threading is available.
units	1..1	int	CPU unit value.
family	1..1	string	CPU family.
model	1..1	string	CPU model.
bogomips	1..1	int	BogoMIPS value.

credentialType

Summary:

Describes the security attributes of a security principle.

Type specification:

Name	Min/Max	Type	Description
`id`	`1..1`	`string`	The ID of the node in the credentials hierarchy.
`policy`	`0..1`	`int`	Policy: `1` -- allow (default) `0` -- deny
`description`	`0..1`	`base64Binary`	Credential description.
`cred`	`0..[]`	`credentialType` (p. 32)	Nested credentials.

eid_listType

Summary:

Holds a list of Server ID values. See `eid_type` (p. 25) for the explanation on what a Server ID is.

Type specification:

Name	Min/Max	Type	Description
`eid`	`0..[]`	`eid_type` (p. 25)	Server IDs.

env_configType

Summary:

Contains server configuration information. This is a base type. It has only the elements that are common to servers of all types (physical and virtual). The subtypes of this type extend it adding more elements that are specific to their respective server types.

Type specification:

Name	Min/Max	Type	Description
`name`	`0..1`	`string`	Server name.
`description`	`0..1`	`base64Binary`	Server description.

domain	0..1	string	Domain name.
hostname	0..1	string	Hostname.
address	0..[]	ip_addressType (p. 42)	List of IP addresses. *Do not use this field when adding or modifying IP addresses of Virtuozzo Containers. Use net_device element of venv_configType (p. 574) instead.*
architecture	0..1	string	CPU architecture.
os	0..1	osType (p. 50)	Operating system.
type	0..1	string	Server type.
nameserver	0..[]	string	Name servers. *Do not use this field when setting nameserver information for Virtuozzo Containers. Use net_device element of venv_configType (p. 574) instead.*
search_domain	0..[]	string	Search domains.
base_sample_id	0..1	guid_type (p. 26)	Base sample config ID.
base_snapshot_id	0..1	guid_type (p. 26)	Base snapshot ID.
child_type	0..[]	string	If a physical or a virtual server hosts other virtual servers, then this element will contain a list of type names of child servers.

Subtypes:

venv_configType (p. 61)

env_resourceType

Summary:

Contains a list of IP addresses allocated to a server.

Type specification:

Name	Min/Max	Type	Description
eid	1..1	eid_type (p. 25)	Server ID.
ip_pool	0..1	ip_poolType (p. 43)	Allocated IP pool.

env_sampleType

Summary:

Base type for the server sample.

Type specification:

Name	Min/Max	Type	Description
`virtual_config`	`0..1`	`venv_configType` (p. 61)	Virtual server configuration information.
`id`	`1..1`	`guid_type` (p. 26)	Global ID of the sample.

env_security_objectType

Summary:

A security object of type "server".

Type specification:

Extends `security_objectType (p. 57)`

Adds the following elements:

Name	Min/Max	Type	Description
`eid`	`1..1`	`eid_type (p. 25)`	Server ID.

env_statusType

Summary:

Contains a server status information.

Type specification:

Name	Min/Max	Type	Description
`state`	`0..1`	`int`	The server state code.
`transition`	`0..1`	`int`	The server transition code.

Description:

A server can be either in a stable state (running, stopped, etc.) or it can be in transition to another stable state (starting, stopping, etc.). For the list of states and transitions, see Appendix B: States and Transitions (p. 656).

envType

Summary:

Contains server information. This type is used for any server type, virtual or physical. However, the `config` and `virtual_config` elements may be instantiated using the server type-specific subtypes of their respective types.

Type specification:

Name	Min/Max	Type	Description
`parent_eid`	`1..1`	`eid_type (p. 25)`	Parent server ID.
`eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`status`	`0..1`	`env_statusType (p. 34)`	Server status.
`alert`	`0..1`	`int`	If any alerts are currently raised on the server then this field will contain the highest existing alert level. See `alert_dataType` (p. 28) for the list of alert levels.
`config`	`0..1`	`env_configType`	Regular configuration information.
`virtual_config`	`0..1`	`env_configType`	Virtual configuration information.

event_dataType

Summary:

The base type defining system event.

Type specification:

The type has no elements.

Subtypes:

`env_status_event_dataType (p. 504)`

`env_config_event_dataType (p. 505)`

`alert_dataType (p. 28)`

resource_alertType (p. 505)

eventType

Summary:

Contains a system event information.

Type specification:

Name	Min/Max	Type	Description
eid	1..1	eid_type (p. 25)	The ID of the server that generated the event.
time	1..1	datetime_type (p. 25)	The time at which the event was generated.
source	1..1	string	The name of the event source -- a plug-in or an operator name.
category	1..1	string	The category of the event.
sid	0..1	sidType (p. 27)	The user SID (security ID). Identifies the active user at the time the event was generated.
count	1..1	int	Message counter. Counts messages received from the same source from the last server restart.
id	1..1	guid_type (p. 26)	A universally unique message ID.
info	1..1	infoType (p. 38)	Event description.
data	0..1		Event type-specific data.
{			
event_data	1..1	event_dataType (p. 35)	Depending on the event type, the actual data type returned will be one of the descendants of event_dataType (p. 35). The data type can be determined by comparing the value of the category element (above) and the event category described in the descendants of event_dataType (p. 35).
}			

Description:

This structure is returned by the calls that provide information about system events and alerts. The elements in the beginning of the structure are common to all event types and provide the basic event information. The `data` element contains the event or alert type-specific data. Depending on the type of the event or alert, the data type of the `event_data` element will be one of the descendants of `event_dataType (p. 35)`. Since you might not know in advance the type of the event, you will have to determine the data type before you can parse the message and handle it properly. Consider the following example.

Let's say that your client program receives an `eventType` structure from Agent as a result of subscription or an on-demand request. Let's also say that the `category` element contains the "env_status" value. If you look at the event type definitions (p. 504), you'll see that "env_status" is the category of the `env_status_event_dataType` (note the `env_status` entry in the Event type subsection). What this means is that in this particular XML response, the data type of the `event_data` element (see table above) is `env_status_event_dataType (p. 504)`, not the base `event_dataType` as shown in the table above.

errorType

Summary:

Informs about the error and provides error details.

Type specification:

Name	Min/Max	Type	Description
`Code`	`0..1`		Unique error ID. It indicates the category to which the error belongs. For details about error classes, see a list of error codes (p. 658).
`Message`	`1..1`		Error summary description.

file_listType

Summary:

Contains a list of file pathnames.

Type specification:

Name	Min/Max	Type	Description
`file_path`	`0..[]`	`string`	A list of pathnames.

groupType

Summary:

User group information structure.

Type specification:

Name	Min/Max	Type	Description
`user`	`0..[]`	`userType (p. 61)`	User info.
`{`			
`  name`	`1..1`	`string`	User name.
`}`			
`member_group`	`0..[]`	`groupType`	Member group info.
`{`			
`  name`	`0..1`	`string`	Group name.
`}`			
`name`	`0..1`	`string`	Group name.
`gid`	`0..1`	`int`	Group ID.

infoType

Summary:

The `infoType` structure is used as a generic container for name/value pairs, and for the text messages that may require localization.

Type specification:

Name	Min/Max	Type	Description
`message`	`1..1`	`base64Binary`	The original message in the official language of the developer. The text may contain references to parameters in the following format: %*param_name*%. The parameter name always begins and ends with a percent sign. The values of the parameters are not included in this element but supplied in the `parameter` element.

translate	0..1	none	If present, indicates that the message contains words in a natural language and, as such, may require a translation to the language of the user.
parameter	0..[]	infoType (p. 38)	The values of the parameters specified in the message element. The value is linked to the parameter using the name element in such a way that the value of the name element of this structure will be the same as the name of the parameter (%*param_name*%) in the message element (see example below). The parameter may also be used as a simple name/value container.
name	1..1	string	Parameter name.

Example:

```
<info>
  <message>T3BlcmF0b3IgJW9wZXJhdG9yJSBhdCAlZWlkJSBzdGFydGVk</message>
  <name></name>
  <translate/>
  <parameter>
    <message>ODQ5YzliZTktNWZiYi00ZTdkLWIxMDAtZjg0MWY4NmMxNTBl</message>
    <name>eid</name>
  </parameter>
  <parameter>
    <message>dnphX2NvbmY=</message>
    <name>operator</name>
  </parameter>
</info>
```

In order to decode the message above, you first have to decode the base64-encoded values that the message contains. The following is the same message with the values decoded to plain text.

```
<info>
  <message>Operator %operator% at %eid% started</message>
  <name></name>
  <translate/>
  <parameter>
    <message>849c9be9-5fbb-4e7d-b100-f841f86c150e</message>
    <name>eid</name>
  </parameter>
  <parameter>
    <message>vzl_conf</message>
    <name>operator</name>
  </parameter>
</info>
```

To process this message, you have to take the following steps:

1 Check if the message possibly requires a translation to the language of the user of your application. For that, you have to check if the `translate` element is present in the packet. In our case, the element is present (which makes sense because the message contains words in a natural language, English in our example), so depending on the target locale, you might want to translate the text portion of it.

2 The second step is to see if the message contains any parameters (the parameter names are preceded by the percent sign %). If there are parameters in the packet, get their values by matching a parameter name to the corresponding name in one of the `parameter` elements that follow the `message` element. In the packet listed above, the first parameter is `%operator%` and its name (`operator`) is contained in the second (from the top) `parameter` element. The value of the parameter is contained in the `parameter/message` element and is `vzl_conf`. The next parameter (`%eid%`) is processed in the same exact manner. If you substitute the parameter references in the original message with their values now, the message will read as follows:

```
Operator vzl_conf at 849c9be9-5fbb-4e7d-b100-f841f86c150e
started.
```

So, the original message that we received essentially means that the `vzl_conf` Agent operator has been started on the server with the Server ID `849c9be9-5fbb-4e7d-b100-f841f86c150e`.

Sometimes the structure is used as a simple generic container for the name/value pairs. For example, the following XML fragment contains information about a server from a backup archive. As you can see, the `message` element in the beginning of the structure does not contain any value, which means that the packet does not contain any message. The underlying parameters simply describe the different properties of a server such as hostname, IP address, operating system, etc.

```
<ns2:info>
  <ns2:message></ns2:message>
  <ns2:name></ns2:name>
  <ns2:translate/>
  <ns2:parameter>
    <ns2:message>SG9zdC0xMDY=</ns2:message>
    <ns2:name>hostname</ns2:name>
    <ns2:translate/>
  </ns2:parameter>
  <ns2:parameter>
    <ns2:message>MTAuMTMwLjEuNg==</ns2:message>
    <ns2:name>ip</ns2:name>
    <ns2:translate/>
  </ns2:parameter>
  <ns2:parameter>
    <ns2:message>VGVzdC1WRTY=</ns2:message>
    <ns2:name>name</ns2:name>
    <ns2:translate/>
  </ns2:parameter>
  <ns2:parameter>
    <ns2:message></ns2:message>
    <ns2:name>os</ns2:name>
    <ns2:parameter>
      <ns2:message>TGludXg=</ns2:message>
      <ns2:name>platform</ns2:name>
    </ns2:parameter>
  </ns2:parameter>
  <ns2:parameter>

<ns2:message>ODllMjc5NjAtOTdiOC00NjFmLTkwMmYtNTU3YjRiMTY3ODRi</ns2:message>
    <ns2:name>parent_eid</ns2:name>
    <ns2:translate/>
  </ns2:parameter>
</ns2:info>
```

interfaceType

Summary:

Describes a network interface.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	Interface name.
`bandwidth`	`0..1`	`int`	Bandwidth.
`transfer`	`0..1`	`transferType (p. 60)`	Transfer rate.

ipaddress	0..1	ip_type (p. 26)	IP address.
flags	0..1	int	Network adapter flags: Bit 0 -- loopback flag. Bit 1 -- no ARP flag.

intervalType

Summary:

A basic date interval structure.

Type specification:

Name	Min/Max	Type	Description
start_time	1..1	datetime_type (p. 25)	Start time.
end_time	1..1	datetime_type (p. 25)	End time.

ip_addressType

Summary:

IP address and netmask.

Type specification:

Name	Min/Max	Type	Description
ip	1..1	ip_type (p. 26)	IP address.
netmask	0..1	ip_type (p. 26)	Netmask.

ip_poolType

Summary:

The IP address information.

Type specification:

Name	Min/Max	Type	Description
[			This is a choice group. Either `ip_range` or `ip` can occur in a document but not both at the same time.
ip_range	1..1		Defines a range of IP addresses.
{			
start_ip	1..1	ip_type (p. 26)	First IP address in the range.
end_ip	1..1	ip_type (p. 26)	Last IP address in the range.
}			
ip	1..1	ip_type (p. 26)	A single IP address.
]			

ip_rangeType

Summary:

IP address range defined by the start IP address and a subnet mask.

Type specification:

Name	Min/Max	Type	Description
id	0..1	string	Range ID.
start_ip	0..1	ip_type (p. 26)	Start IP address.
subnet_mask	0..1	int	Subnet mask.
comment	0..1	string	Comments.

load_avg_statsType

Summary:

Contains load average statistics.

Type specification:

Name	Min/Max	Type	Description
l1	1..1	statsType	1 minute Load Average values.
{			
avg	0..1	long	Average value.
min	0..1	long	Minimum value.
max	0..1	long	Maximum value.
cur	0..1	long	Current value.
}			
l2	0..1	statsType	5 minute Load Average values.
{			
avg	0..1	long	Average value.
min	0..1	long	Minimum value.
max	0..1	long	Maximum value.
cur	0..1	long	Current value.
}			
l3	0..1	statsType	15 minute Load Average values.
{			
avg	0..1	long	Average value.
min	0..1	long	Minimum value.
max	0..1	long	Maximum value.
cur	0..1	long	Current value.
}			

load_avgType

Summary:

Load average values.

Type specification:

Name	Min/Max	Type	Description
`l1`	`1..1`	`double`	1 minute Load Average value.
`l2`	`0..1`	`double`	5 minute Load Average value.
`l3`	`0..1`	`double`	15 minute Load Average value.

log_options_baseType

Summary:

Base type.

Type specification:

The type has no elements.

Subtypes:

`log_optionsType`

`log_optionsType` (p. 568)

log_optionsType

Summary:

The type for logging options.

Type specification:

Extends `log_options_baseType (p. 45)`

The type has no additional elements.

modType

Summary:

The `modType` type is used when modifying a user or a group. It is populated with the list of the user/group attributes and the type of the modification to perform on them.

Type specification:

Extends `named_listType (p. 46)`

Adds the following elements:

Name	Min/Max	Type	Description
`op`	`0..1`	`int`	Modification type: `0` -- Add the specified attribute values (default). `1` -- Delete the specified attribute values. `2` -- Replace the existing attribute values with the values specified. If an attribute has more than one value, all of those values will be removed and the new values will be used in their place.

named_listType

Summary:

Attribute name/value pair.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	Attribute name.
`value`	`0..[]`	`base64Binary`	Attribute value(s).

native_configType

Summary:

The base type for the virtual server configuration data in a corresponding virtualization technology's native format. Agent uses its own structures (types) for the virtual server configuration data. Server virtualization products, such as Virtuozzo Containers, cannot directly use this data because they don't natively understand the format. The subtypes of `native_configType` are used to hold the configuration data in a format understood by a specific virtualization technology. While Agent configuration structures make sense within the context of Agent, the native configuration data can be used externally. For example, you can pass it to a utility that comes with your virtualization product and which expects it as an input. The `envm` interface (p. 193) provides two calls that allow to convert the configuration data between the two formats -- `get_native_config` (p. 228) and `get_virtual_config`.

Note: At the time of this writing, the only supported virtualization technology is Virtuozzo Containers.

Type specification:

The type has no elements.

Subtypes:

`virtuozzo_configType` (p. 577)

net_addressType

Summary:

Network address.

Type specification:

Name	Min/Max	Type	Description
`host`	`1..1`	`none`	Host/Net name or IP address.
`mask`	`0..1`	`ip_type (p. 26)`	IP mask.

net_classType

Summary:

Network class.

Type specification:

Name	Min/Max	Type	Description
id	0..1	string	Class ID.
transfer	0..1	transferType (p. 60)	Transfer statistics.

net_deviceType

Summary:

Holds a generic network device information. A networks device can be a network interface card, a network bridge, or a virtual local area network. The subtypes of this type extend it adding additional functionality.

Type specification:

Name	Min/Max	Type	Description
id	0..1	string	Device ID (e.g. interface name eth0, eth1, br2, etc).
ip_address	0..[]	ip_addressType (p. 42)	The list of IP addresses assigned to this device.
dhcp	0..1	none	DHCP flag. If present, the device uses DHCP to receive TCP/IP settings.
network_id	0..1	base64Binary	The ID of the virtual network this device belongs to.
status	0..1		Network device status.
[			Denotes a choice between the up and the down elements.
up	1..1	none	The device is up (enabled).
down	1..1	none	The device is down (disabled).
]			

Subtypes:

net_nicType (p. 49)

net_vethType (p. 569)

net_vlanType (p. 295)

net_bridgeType (p. 296)

net_nicType

Summary:

Contains a physical network adapter information.

Type specification:

Extends net_deviceType (p. 48)

Adds the following elements:

Name	Min/Max	Type	Description
mac_address	0..1	string	MAC address.

operator_functionalType

Summary:

Agent Operators are represented in the front-end API by *interfaces*. Each interface provides methods for accessing a corresponding operator functionality. Each interface is identified by an XML element to be used in the XML request sent to Agent server. Each such element is derived from type operator_functionalType. The type defines the basic input and output parameters. Interfaces extend it with their own input and output parameters.

Type specification:

Name	Min/Max	Type	Description
ok	1..1		This element is returned to the client when a request completes successfully but does not return any information by definition.
error	1..1		This element is returned when a request results in failure.
{			

code	1..1	int	Error code.
message	0..1	string	Text message describing the error.
}			

osType

Summary:

Contains the operating system information.

Type specification:

Name	Min/Max	Type	Description
platform	0..1	string	OS platform (Windows, Linux, etc.).
name	1..1	string	OS name (Windows XP, Red Hat 9, etc.).
version	0..1	string	OS version.
kernel	0..1	string	Kernel ID.

packageType

Summary:

A generic software package information. Agent supports different types of software packages, including different varieties of the Linux specific packages, and Virtuozzo Containers specific packages. When the packageType type is used as an input parameter, use one of the appropriate subtypes depending on the type of the software package. When it is used as an output, expect the appropriate subtype in the response packet.

Type specification:

Name	Min/Max	Type	Description
name	1..1	string	Package name.
summary	0..1	string	Summary description.
os	0..1	osType (p. 50)	Target OS and platform.
description	0..1	string	Description.
arch	0..1	string	Package architecture -- x86, x86_64, etc.
version	0..1	string	Package version.

Subtypes:

package_linuxType (p. 318)

package_rpmType (p. 319)

package_debType (p. 318)

package_vztemplateType (p. 572)

package_std_vztemplateType (p. 572)

perf_dataType

Summary:

Contains data returned by the performance monitor.

Type specification:

Name	Min/Max	Type	Description
eid	1..1	eid_type (p. 25)	Server ID.
class	0..[]	none	A list of performance classes that were selected for monitoring.
{			
name	1..1	string	Class name.
instance	1..[]	none	A list of class instances that were selected for monitoring.
{			
name	0..1	string	Instance name.
counter	1..[]	none	A list of performance counters that were selected for monitoring.
{			
name	1..1	string	Counter name.
value	1..1	perf_statType (p. 52)	Values.
}			
}			
}			
interval	1..1	intervalType (p. 42)	The time interval over which the data was collected.

perf_statType

Summary:

Performance statistics.

Type specification:

Name	Min/Max	Type	Description
cur	1..1	anySimpleType	Current value.
avg	1..1	anySimpleType	Average value.
max	1..1	anySimpleType	Maximum value.
min	1..1	anySimpleType	Minimum value.

policy_ruleType

Summary:

Contains a policy rule information.

Type specification:

Name	Min/Max	Type	Description
id	1..1	string	Rule ID.
value	0..1	base64Binary	Rule value. If not specified, the default value is used.

processesType

Summary:

Contains information about system processes.

Type specification:

Name	Min/Max	Type	Description
run	1..1	int	Number of processes in a "running" state.
zombie	1..1	int	Number of processes in a "zombie" state.
sleep	1..1	int	Number of processes in a "sleep" state.
uninterrupt	1..1	int	Number of processes in a "uninterrupt" state.

stopped	1..1	int	Number of processes in a "stopped" state.
total	1..1	int	Total number of processes.

ps_infoType

Summary:

Contains information about system processes.

Type specification:

Name	Min/Max	Type	Description
process	1..[]	none	Process information
{			
pid	1..1	int	Process identifier (PID).
param	0..[]	base64Binary	A list of process parameter values in the order indicated by the list contained in the param_id element.
}			
param_id	1..[]	string	A list of the included process parameters in the order in which their values appear in the param element.

qosType

Summary:

Contains QoS (quality of service) limits.

Type specification:

Name	Min/Max	Type	Description
id	1..1	string	QoS counter ID.
soft	0..1	long	Soft limit.
hard	0..1	long	Hard limit.
cur	0..1	long	Current value.

realmType

Summary:

The base type defining a Realm. A Realm is an authentication database containing Parallels Infrastructure Management user and group information. Agent supports a number of different databases, including operating system user registries and LDAP-compliant directories. Realm definitions are stored in the Agent configuration and can be retrieved using the `get_realm` call (p. 99).

Type specification:

Name	Min/Max	Type	Description
`id`	`0..1`	`guid_type` (p. 26)	Globally unique Realm ID. The ID is automatically generated by Agent when a Realm definition is added to the Agent configuration. The ID is guaranteed to be unique across different computers and networks.
`type`	`1..1`	`int`	Realm type: `0` -- System Realm. This is the operating system user registry on the Hardware Node. `1` -- LDAP directory Realm. This is an LDAP-compliant directory such as AD or ADAM on Windows, or OpenLDAP on Linux. The directory can be located locally or anywhere on the network. `1000` -- Virtuozzo Container Realm. This is the operating system user registry inside a Virtuozzo Container.
`name`	`1..1`	`string`	The name of the Realm (user defined).
`builtin`	`0..1`		If present, indicates that this is a built-in Realm. A built-in Realms is a preinstalled authentication database. It can be an LDAP directory or an operating system user registry. Built-in Realms contain Parallels Infrastructure Management authentication and authorization information, including built-in security roles, permissions, users, and groups used by Virtuozzo Tools.

Subtypes:

`dir_realmType` (p. 73)

resourceType

Summary:

A generic type for specifying a resource information.

Type specification:

Name	Min/Max	Type	Description
`total`	`0..1`	`long`	Total units available.
`used`	`0..1`	`long`	Number of units currently used.
`free`	`0..1`	`long`	Number if units currently available.
`avg`	`0..1`	`long`	Average usage.
`min`	`0..1`	`long`	Minimum units used.
`max`	`0..1`	`long`	Maximum units used.

sample_confType

Summary:

Sample configuration information.

Type specification:

Name	Min/Max	Type	Description
`env_config`	`1..1`	`env_configType`	Configuration parameters.
`id`	`0..1`	`guid_type` (p. 26)	Sample configuration ID.
`name`	`1..1`	`string`	Sample configuration name.
`comment`	`0..1`	`base64Binary`	Comment.
	`1..1`		
`vt_version`	`0..1`		Virtualization technology information.
`{`			
`platform`	`1..1`	`string`	Platform (WIndows, Linux, etc.).
`architecture`	`1..1`	`string`	Architecture.

vt_technology	1..1	string	Virtualization technology name.
}			

security_descriptorType

Summary:

Security descriptor.

Type specification:

Name	Min/Max	Type	Description
owner	1..1	sidType (p. 27)	Security ID of a security object owner. The owner is the user that is always allowed to control the DACL of the object.
group	1..1	sidType (p. 27)	Group security ID. It is used as a way of tracking a group for each object providing support for Linux permissions.
dacl	0..1		Discretionary Access Control List (DACL).
{			
ace	0..[]	aceType (p. 27)	Access control entry.
}			

Description:

Each object protected by the Agent access control system must have a state associated with it to track its security settings. This state is called *security descriptor*.

The discretionary access control list (DACL) contains a list of permissions granted or denied to various users and groups. The owner of the object is always allowed to control the DACL contents.

The access control entry (ACE) is an individual record in a DACL. It includes the SID of a single user or a group along with an access mask that specifies the permissions being granted or denied.

security_objectType

Summary:

Base type describing a security object. Implemented in descendants.

Type specification:

The type has no elements.

statsType

Summary:

Holds QoS data.

Type specification:

Name	Min/Max	Type	Description
`avg`	`0..1`	`long`	Average value
`min`	`0..1`	`long`	Minimum value
`max`	`0..1`	`long`	Maximum value
`total`	`0..1`	`long`	Total value
`cur`	`0..1`	`long`	Current value
`soft`	`0..1`	`long`	Soft limit
`hard`	`0..1`	`long`	Hard limit

sys_infoType

Summary:

System information.

Type specification:

Name	Min/Max	Type	Description
load_avg	1..1	load_avgType (p. 45)	Load averages
processes	1..1	processesType (p. 52)	Processes statistics
cpu_load	1..1	cpu_loadType (p. 31)	CPU load statistics.
cpu_states	1..1	cpu_loadType (p. 31)	CPU load statistics (percentage).
users	1..1	int	Number of users.
uptime	1..1	long	Uptime.
memory	0..1	resourceType (p. 55)	Memory statistics. Provided only for Hardware Node.
{			
total	0..1	long	Total memory available
used	1..1	long	Memory used.
}			
swap	0..1	resourceType (p. 55)	Swap statistics. Provided only for Hardware Node.
{			
total	0..1	long	Total.
used	1..1	long	Used.
}			

system_nodeType

Summary:

Contains computer access parameters.

Type specification:

Name	Min/Max	Type	Description
address	1..1		IP address.
{			

ip	1..1	ip_type (p. 26)	IP address.
}			
login	0..1		Login info.
{			
user	1..1	string	User name.
password	1..1	base64Binary	Password.
}			

tokenType

Summary:

Security token.

Type specification:

Name	Min/Max	Type	Description
user	1..1	sidType (p. 27)	SID of the token owner.
groups	0..1		The list of SIDs of the groups to which the token owner belongs.
{			
sid	1..[]	sidType (p. 27)	SIDs.
}			
deny_only_sids	0..1		The list of deny-only SIDs.
{			
sid	0..[]		SIDs.
}			
privileges	0..1		*These fields are not currently used.*
{			
privilege	0..[]	privilegeType (p. 26)	
}			
source	0..1		*These fields are not currently used.*
{			
name	1..1	string	
id	1..1	guid_type	

}			

transferType

Summary:

Contains network transfer statistics.

Type specification:

Name	Min/Max	Type	Description
input	1..1	none	Incoming traffic.
{			
bytes	1..1	long	Number of bytes.
packets	0..1	long	Number of packets.
}			
output		none	Outgoing traffic.
{			
bytes	1..1	long	Number of bytes.
packets	0..1	long	Number of packets.
}			

usageType

Summary:

A generic type for specifying a resource usage information.

Type specification:

Name	Min/Max	Type	Description
total	0..1	long	Total units.
used	0..1	long	Used units.
free	0..1	long	Free units.

userType

Summary:

User information.

Type specification:

Name	Min/Max	Type	Description
initial_group	0..1		Initial group
{			
name	0..1	string	Group name
gid	0..1	int	Group ID
}			
group	0..[]		Other groups.
{			
name	0..1	string	Group name.
gid	0..1	int	Group ID.
}			
uid	0..1	int	User ID.
shell	0..1	string	Shell.
password	0..1	base64Binary	Password.
home_dir	0..1	string	Home directory.
name	0..1	string	User name.
comment	0..1	string	Comment.

venv_configType

Summary:

Virtual server configuration. Virtuozzo-specific structure is derived from this type and contains additional parameters.

Type specification:

Extends env_configType

Adds the following elements:

Name	Min/Max	Type	Description
qos	0..[]	qosType (p. 53)	QoS parameters.

Subtypes:

`vzat:venv_configType` (p. 574)

voc_parameterType

Summary:

Contains an Agent vocabulary entry information.

Type specification:

Name	Min/Max	Type	Description
`id`	`1..1`	`string`	Vocabulary entry ID.
`type`	`0..1`	`string`	Data type (int, string, etc.)
`min`	`0..1`	`string`	Minimum possible value.
`max`	`0..1`	`string`	Maximum possible value.
`long`	`0..1`	`string`	Long description.
`short`	`0..1`	`string`	Short description.
`category`	`0..[]`	`string`	Category.
`complex`	`0..1`	`string`	Entry type-specific value. May have different meaning for different types of entries.
`default`	`0..1`	`string`	Default value.
`measure`	`0..1`	`string`	Units of measure.
`data`	`0..1`		Data (entry-type specific value).
`name`	`0..1`	`string`	Entry name.

vocabularyType

Summary:

Contains the Agent vocabulary data.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	The name of Agent plug-in that this vocabulary is describing.
`parameter`	`0..[]`	`voc_parameterType (p. 514)`	The vocabulary parameters.
`category`	`0..[]`	`voc_parameterType (p. 514)`	The vocabulary categories.

vt_infoType

Summary:

Virtualization technology-specific read-only settings.

Type specification:

Name	Min/Max	Type	Description
`xs:any`			

vt_settingsType

Summary:

A base type defining virtualization technology-specific settings. See subtypes for implementations.

Type specification:

Name	Min/Max	Type	Description
`default_sample_id`	`0..1`	`guid_type` (p. 26)	The default sample configuration ID.

Subtypes:

`vzat.vt_settingsType` (p. 578)

Interfaces

The material in this section describes the base Agent XML API interfaces. The term *interface*, as we use it, is somewhat similar to a class in object-oriented programming. We use interfaces to group related data types (structures) and calls (methods). The data types and calls are defined using XML Schema language (XSD). The body of an Agent XML request always begins with the name of an interface followed by the name of a call. The rest of the request body is composed according to the call specifications.

The base interfaces described in this chapter form a foundation for the Agent XML API and currently provide functionality for the Hardware Node and Virtuozzo Containers management. Some of the Virtuozzo-specific functionality exists as a plug-in consisting of additional interfaces, some of which are derived from the base interfaces. Virtuozzo Containers plug-in is described in the Virtuozzo Containers Types and Interfaces chapter (p. 567).

alertm

Purpose:

The `alertm` interface allows the user to receive notifications on critical system events via e-mail and to retrieve the list of currently raised alerts.

Types

resource_alertType

Summary:

Resource allocation alert data.

Type specification:

Extends `alert_dataType (p. 28)`

Adds the following elements:

Name	Min/Max	Type	Description
`eid`	`0..1`	`eid_type (p. 25)`	Server ID.
`class`	`1..1`	`string`	Performance class (see `perf_mon` (p. 352) for more info on this and the following parameters).
`instance`	`1..1`	`string`	Class instance.

counter	1..1	string	Performance counter.
cur	1..1	anySimpleType	Current value.
hard	1..1	anySimpleType	Hard value.
soft	1..1	anySimpleType	Soft value.

server_group_alertType

Summary:

Virtuozzo group-wide alert data.

Type specification:

Extends alert_dataType (p. 28)

Adds the following elements:

Name	Min/Max	Type	Description
eid	1..1	eid_type (p. 25)	The ID of the Container that caused the alert.
address	1..1	string	The address of the slave node hosting the Container specified in the eid element (above).
title	1..1	string	The title of the slave node.
description	1..1	string	Problem description.
code	1..1	string	Error code.

Calls

Call	Description
get_alerts (p. 66)	Returns current alert states for the specified list of servers.
subscribe_alert (p. 68)	Subscribes to receive alert notifications via e-mail.
unsubscribe_alert (p. 71)	Cancels alert subscriptions.

get_alerts

Summary:

Returns current alert states for specified servers.

Request specification:

Name	Min/Max	Type	Description
`get_alerts`			
`{`			
`eid_list`	`0..1`	`eid_listType` (p. 32)	Server list.
`category`	`0..1`	`string`	Categories of the alerts to return. If this element is omitted, the call will return alerts of all known types. The available alert categories are: ▪ `resource_alert` ▪ `server_group_alert`
`env_type`	`0..1`	`string`	Return alerts only for the servers of the specified type (`generic`, `virtuozzo`).
`}`			

Returns:

Name	Min/Max	Type	Description
`alert`	`0..[]`	`eventType` (p. 36)	Alert information.

Description:

A server may have multiple alerts of different types raised at any given time. You can use subscriptions to receive alert notifications in real time (see `subscribe` (p. 558) and `subscribe_alert` (p. 68)), or you can retrieve all alerts that are currently raised on any given server using the `get_alerts` call.

Example:

The following sample shows how to obtain states of alerts for the specified server(s).

Input

```
<packet>
  <target>alertm</target>
  <data>
    <alertm>
      <get_alerts>
        <eid_list>
          <eid>ccc794ad-cc5d-49f2-8d84-6631263c81be</eid>
        </eid_list>
      </get_alerts>
    </alertm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/alertm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="2dc4ad4739ft26e9rec4" time="2009-10-13T12:33:17+0000">
<ns1:origin>alertm</ns1:origin>
<ns1:target>vzclient763-75e18434-5abc-5db7-e963-9f67a2f03412</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:alertm>
        <ns2:alert xsi:type="ns3:eventType">
            <ns3:eid>75e18434-5abc-5db7-e963-9f67a2f03412</ns3:eid>
            <ns3:time>2009-10-09T02:58:33+0000</ns3:time>
            <ns3:source>ResourceAlertMonitor</ns3:source>
            <ns3:category>resource_alert</ns3:category>
            <ns3:sid>AQUAAAAAIAM0hOF1vFq3Xeljn2ei8DQSAAAAAA==</ns3:sid>
            <ns3:count>1</ns3:count>
            <ns3:id>83c11a69-5c1f-4836-abc0-de2d779f3331</ns3:id>
            <ns3:data>
                <ns3:event_data xsi:type="ns2:resource_alertType">
                    <ns2:eid>75e18434-5abc-5db7-e963-9f67a2f03412</ns2:eid>
                    <ns2:type>1</ns2:type>
                    <ns2:class>counters_disk</ns2:class>
                    <ns2:instance>\\?\Volume{388c3805-7509-11de-8c64-
806e6f6e6963}\</ns2:instance>
                    <ns2:counter>counter_disk_share_used</ns2:counter>
                    <ns2:cur>79</ns2:cur>
                    <ns2:soft>85</ns2:soft>
                    <ns2:hard>95</ns2:hard>
                </ns3:event_data>
            </ns3:data>
            <ns3:info xsi:type="ns3:infoType">
```

```
<ns3:message>UmVzb3VyY2UgJWlkJSAldHlwZSUgYWxlcnQgb24gZW52aXJvbm1lbnQgJWVudiUgY
3VycmVudCB2YWx1ZTogJWN1ciUgc29mdCBsaW1pdDogJXNvZnQlIGhhcmQgbGltaXQ6ICVoYXJkJQ=
=</ns3:message>
                <ns3:name></ns3:name>
                <ns3:translate/>
                <ns3:parameter>
                    <ns3:message>Nzk=</ns3:message>
                    <ns3:name>cur</ns3:name>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>JXRpdGxlJQ==</ns3:message>
                    <ns3:name>env</ns3:name>
                    <ns3:translate/>
                    <ns3:parameter>

<ns3:message>NzVlMTg0MzQtNWFiYy01ZGI3LWU5NjMtOWY2N2EyZjAzNDEy</ns3:message>
                        <ns3:name>eid</ns3:name>
                        <ns3:translate/>
                    </ns3:parameter>
                    <ns3:parameter>
                        <ns3:message>RVZCWU1JTlNEMDIyMw==</ns3:message>
                        <ns3:name>title</ns3:name>
                        <ns3:translate/>
                    </ns3:parameter>
                    <ns3:parameter>
                        <ns3:message>Z2VuZXJpYw==</ns3:message>
                        <ns3:name>type</ns3:name>
                        <ns3:translate/>
                    </ns3:parameter>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>OTU=</ns3:message>
                    <ns3:name>hard</ns3:name>
                </ns3:parameter>
                <ns3:parameter>

<ns3:message>Y291bnRlcl9kaXNrX3NoYXJlX3VzZWQ=</ns3:message>
                    <ns3:name>id</ns3:name>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>ODU=</ns3:message>
                    <ns3:name>soft</ns3:name>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>eWVsbG93</ns3:message>
                    <ns3:name>type</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
            </ns3:info>
        </ns2:alert>
    </ns2:alertm>
</ns1:data>
```

subscribe_alert

Summary:

Subscribes to alert notifications via e-mail.

Request specification:

Name	Min/Max	Type	Description
`subscribe_alert`			

{			
eid_list	0..1	eid_listType (p. 32)	Server IDs. If omitted, subscribes to receive alert notifications for all known servers.
email	1..1	string	The e-mail address to send the notifications to.
name	0..1	string	The name of the e-mail template. The template is configured using the mailer interface (p. 287). If not specified, the default template for the specific alert type will be used.
[	0..1		This section specifies the alert type. If the section is omitted, subscribes to QoS alerts by default.
services	1..1		Get alerts on changing service status.
{			
service	1..[]	string	Service name.
}			
]			
}			

Returns:

OK/Error.

Description:

To prevent alert (mail) flooding you can set `mute_alert_period` configuration parameter in the `alertm` section of Agent configuration. Negative value means that subscription stops after the first alert and you have to re-subscribe. Zero value turns off flooding control, i.e. all alerts will be delivered. Positive value means that subsequent alerts for the same servers will be delivered at once in case of period expiration or if alert level is greater than the one of the previous alert.

To set a default e-mail address, use the `support_email` parameter in the `alertm` section of Agent configuration.

To receive alert notifications directly (not through e-mail), use the `system/subscribe` call (p. 558) together with `resource_alert` event (p. 511).

To unsubscribe from this service, use the `unsubscribe_alert` call (p. 71).

Example:

Input

```
<packet>
  <target>alertm</target>
  <data>
    <alertm>
      <subscribe_alert>
        <eid_list>
          <eid>ccc794ad-cc5d-49f2-8d84-6631263c81be</eid>
        </eid_list>
        <email>johndoe@mail.com</email>
        <services>
          <service>crond</service>
        </services>
      </subscribe_alert>
    </alertm>
  </data>
</packet>
```

Output

```
<packet>
  <origin>alertm</origin>
  <data>
    <alertm>
      <ok/>
    </alertm>
  </data>
</packet>
```

Incoming email

```
From:  support@tc6.com
To:    johndoe@mail.com
Subject:     Service crond is stopped on ccc794ad-cc5d-49f2-8d84-6631263c81be
at 2006-05-06T19:04:01+0000
       Service crond changed status to stopped on ccc794ad-cc5d-49f2-8d84-
6631263c81be at 2006-05-06T19:04:01+0000
```

unsubscribe_alert

Summary:

Cancels an alert notification subscription.

Request specification:

Name	Min/Max	Type	Description
`unsubscribe_alert`			
`{`			
`eid_list`	`0..1`	`eid_listType (p. 32)`	Server ID list.
`email`	`1..1`	`string`	E-mail that was used to subscribe for alerts.
`services`	`1..1`		Cancel service status changes alerts.

Returns:

OK/Error

Description

Use the `unsubscribe_alert` call if you would like to stop receiving alert notifications through e-mail that you previously subscribed to using the `subscribe_alert` call (p. 68).

Example:

The following sample shows how to obtain states of alerts for the specified server(s).

Input

```
<packet>
  <target>alertm</target>
  <data>
    <alertm>
      <unsubscribe_alert>
        <eid_list>
          <eid>a6f1d061-8bcd-8ec7-1a73-b078fe2d416f</eid>
        </eid_list>
        <email>siarhei_rabtsau@epaml.com</email>
        <services>
        <service>crond</service>
        </services>
      </unsubscribe_alert>
    </alertm>
</data>
</packet>
```

Output

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/alertm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="15c4adf08b8t5f90r2c8" time="2009-10-21T13:10:49+0000">
<ns1:origin>alertm</ns1:origin>
<ns1:target>vzclient8271-a6f1d061-8bcd-8ec7-1a73-b078fe2d416f</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:alertm>
        <ns1:error>
            <ns1:code>4102</ns1:code>
            <ns1:message>Unsubscribe error: Subscription not found email:
siarhei_rabtsau@epaml.com, eid: a6f1d061-8bcd-8ec7-1a73-
b078fe2d416f</ns1:message>
        </ns1:error>
    </ns2:alertm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

authm

Purpose:

User/group profile management, user authentication, realm management.

Types

dir_realmType

Summary:

Defines an LDAP directory-based realm (an LDAP-compliant directory such as AD or ADAM on Windows, or OpenLDAP on Linux).

Type specification:

Extends realmType (p. 54).

Adds the following elements:

Name	Min/Max	Type	Description
`address`	`1..1`	`string`	The IP address or the hostname of the server hosting the directory.
`port`	`1..1`	`int`	The TCP port at which the directory instance is listening for requests.
`base_dn`	`1..1`	`string`	Base DN. This is the top level of the directory tree.
`default_dn`	`1..1`	`string`	Default DN. This is the distinguished name of the default container where the user information is stored. The default DN allows you to pass the user name during login or other operations as a plain name instead of using a fully qualified DN. For example, let's say that the default DN is `CN=Users,DC=Mydomain` If the user name is passed as `TestUser` (for example), then Agent will automatically construct a full DN by adding the `TestUser` to the default DN.
`login`	`0..1`	`loginType`	Login information. This information is used to establish a connection with the directory instance to perform authentication. The user specified here must exist in the directory and should have sufficient rights to use the directory services.

security_principalType

Summary:

Contains information about a security principal.

Type specification:

Extends `auth_nameType (p. 29)`

Adds the following elements:

Name	Min/Max	Type	Description
`group`	`0..[]`	`auth_nameType (p. 29)`	A list of groups to which this user or group belongs. When adding a user or a group, this element is used to specify the groups to add the principal user/group to.
`member_group`	`0..[]`	`auth_nameType (p. 29)`	A list of member groups.
`member_user`	`0..[]`	`auth_nameType (p. 29)`	A list ot member users.
`data`	`0..1`		The user/group profile data.
`{`			
`  attr`	`0..[]`	`named_listType (p. 46)`	Attributes and their values. The available attributes are determined by the "database" used. In an LDAP directory these would be the attributes of an object representing the user.
`}`			

sp_modType

Summary:

Used when modifying a user or a group.

Type specification:

Name	Min/Max	Type	Description
`name`	`0..1`	`base64Binary`	User/group name.
`domain`	`0..1`	`base64Binary`	Domain name.
`data`	`0..1`		Attributes.

{			
mod	0..[]	modType (p. 46)	A list of the user/group attributes. The values can be added, deleted, or replaced based on the specified operation type. See modType (p. 46) for details.
}			

Calls

Call	Description
add_group (p. 75)	Adds a new group to the specified realm.
add_user (p. 82)	Adds a new user to the specified realm.
edit_group (p. 498)	Modifies an existing group.
edit_user (p. 92)	Modifies an existing user.
add_to_group (p. 80)	Adds a user/group to other groups as a member.
del_from_group (p. 86)	Deletes a user/group from groups.
get_group (p. 496)	Retrieves group information.
get_user (p. 492)	Retrieves user information.
del_group (p. 489)	Deletes a group from the database.
del_user (p. 89)	Deletes a user from the database.
authenticate (p. 84)	Authenticates a user.
add_realm (p. 78)	Adds a new Realm definition to the Agent configuration.
del_realm (p. 88)	Deletes an existing Realm definition.
get_realm (p. 99)	Retrieves information about existing realms.
set_realm (p. 105)	Modifies the specified realm definition.
get_auth_name (p. 94)	Returns login information for the security account specified by the SID.
get_sid (p. 101)	Returns SID of the security account specified by the login info.

add_group

Summary:

Adds a new group to the specified Realm.

Request specification:

Name	Min/Max	Type	Description
add_group			

{			
group	1..1	security_principalType (p. 74)	The new group information.
}			

Returns:

OK/Error

Description:

The group name can be specified as a DN or as a plain name. If you specify a fully qualified distinguished name, the group will be created in the specified location in the directory tree. If you use a plain name, the group will be created using the specified name in the default container for this realm. To find out what the default DN is, use the `get_realm` call (p. 99).

Note: The call will try to create a new directory object of class `Group` (`objectClass=Group`). Your directory schema must have the `Group` class in it or the call will fail.

Example:

Creating a new group named `Test_Group` in the specified Realm. The group name is specified as a plain name so it will be created in the default container for this Realm.

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <add_group>
        <group>
          <name>VGVzdF9Hcm91cA==</name>
          <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
          <data>
            <attr>
              <name>description</name>
              <value>VGhpcyBpcyBhIHRlc3QgZ3JvdXA=</value>
            </attr>
          </data>
        </group>
      </add_group>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="9c469dcebft6784r4b4"
time="2007-07-18T06:21:21+0000" priority="0" version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient4-2cbbe469-8f57-7f46-97fb-fd987231d957</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns1:ok/>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
```

```
</ns1:src>
</packet>
```

add_realm

Summary:

Adds a Realm definition to the Agent configuration.

Request specification:

Name	Min/Max	Type	Description
add_realm			
{			
realm	1..1	realmType (p. 54)	Realm information. Depending on the realm type, use the appropriate subtype of the realmType type (p. 54).
password	0..1	base64Binary	The password for the user account specified in the login portion of the realm structure above.
}			

Returns:

Name	Min/Max	Type	Description
id	1..1	guid_type	The new realm ID (automatically generated by Agent). The ID is universally unique.

Description:

When adding an LDAP directory realm, the call does not verify whether the values that you supply are valid or not. It verifies the basic syntax, but it doesn't actually try to connect to the directory. After you execute the call, you should check that you can connect to the directory and retrieve the data from it. For example, you can try getting a user information from with the `get_user` call (p. 102).

Note: When adding an LDAP directory Realm please make sure that the users in your directory are stored as objects of type `User` (`objectClass=User`) and that the groups are stored as objects of type `Group` (`objectClass=Group`). If the user and group objects use different classes, you will not be able to see or authenticate them in Agent.

Example:

The following example shows how to create a typical LDAP directory realm. The following table describes the parameters and their values used in the example.

Parameter	Value	Description
`type`	`1`	LDAP directory realm.
`name`	`myrealm`	Realm name.
`address`	`192.168.0.117`	The IP address of the server hosting the LDAP directory.
`port`	`398`	TCP port number on which the directory instance is listening for requests.
`base_dn`	`dc=vzl`	Base DN (the top level of the directory tree).
`default_dn`	`cn=users,dc=vzl`	Default DN (the default container in the directory where the user information is stored).
`login/name`	`cn=vzagent,dc=VZL`	The DN of the user that will be used to connect to the directory instance to perform authentications.
`password`	`bXlwYXNz`	The password for the user specified in the `login/name` parameter above.

Input

```
<packet xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <add_realm>
        <realm xsi:type="ns4:dir_realmType">
           <type>1</type>
           <name>myrealm</name>
           <address>192.168.0.117</address>
           <port>398</port>
           <base_dn>dc=vzl</base_dn>
           <default_dn>cn=users,dc=vzl</default_dn>
           <login>
             <name>Y249dnphZ2VudCxkYz1WWkw=</name>
           </login>
         </realm>
          <password>bXlwYXNz</password>
      </add_realm>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="8c460a8b7ft6952r350"
time="2007-03-21T12:21:57+0000" priority="0" version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns2:id>18e8c5f0-e656-4144-864c-0520275a4bd1</ns2:id>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

add_to_group

Summary:

Add a user or a group to other groups as a member.

Request specification:

Name	Min/Max	Type	Description
add_to_group	1..1	auth_nameType (p. 29)	The inherited auth_nameType portion is used to specify the user/group to add to other groups as a member.
{			

group	1..[]	auth_nameType (p. 29)	Parent groups. The specified user or group will be added as a member to these groups.
}			

Returns:

OK/Errors.

Description:

The call adds a user or a group to another group or to multiple groups. This call works only with the users and groups that already exist. To create a new user or a group and add it to another group at the same time, use the add_user (p. 82) or add_group (p. 75) calls.

Example:

Adding a user Test_User from the default realm to the Administrators group.

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <add_to_group>
        <name>VGVzdF9Vc2Vy</name>
        <realm>00000000-0000-0000-0000-000000000005</realm>
        <group>
          <name>VGVzdF9Hcm91cA==</name>
          <realm>00000000-0000-0000-0000-000000000005</realm>
        </group>
      </add_to_group>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="36c4ae00f60t4dc8r2c8" time="2009-10-22T07:53:34+0000">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient60-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:authm>
        <ns1:ok/>
    </ns2:authm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

add_user

Summary:

Adds a new user to the specified realm.

Request specification:

Name	Min/Max	Type	Description
`add_user`			
`{`			
`user`	`1..1`	`security_principalType (p. 74)`	The new user information.
`password`	`1..1`	`base64Binary`	A password to set for the new user.
`}`			

Returns:

OK/Error

Description:

The user name can be specified as a distinguished name (DN) or as a plain name. If you use a fully qualified DN, the user will be created in the specified location in the directory tree. If you use just a plain name, the user will be created in the default container for this realm. To find out what the default DN is, use the `get_realm` call (p. 99).

Note: The call will try to create a new directory object of class `User` (`objectClass=User`). Your directory schema must have the `User` class in it or the call will fail.

Example:

Creating a new user named `Test_User` in the specified realm. The user name is specified as a plain name so it will be created in the default container for this Realm.

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <add_user>
        <user>
          <name>VGVzdF9Vc2Vy</name>
          <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
          <data>
            <attr>
              <name>description</name>
              <value>VGhpcyBpcyBhIHRlc3QgdXNlcg==</value>
            </attr>
          </data>
        </user>
        <password>bXlwYXNzd2Q=</password>
      </add_user>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="23c469e077at4db7r4b4" time="2007-07-18T08:57:51+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient4-2cbbe469-8f57-7f46-97fb-fd987231d957</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns1:ok/>
  </ns2:authm>
</ns1:data>
```

```
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</packet>
```

authenticate

Summary:

Authenticates the specified user.

Request specification:

Name	Min/Max	Type	Description
`authenticate`	`1..1`	`auth_nameType` (p. 29)	The inherited `auth_nameType` portion is used to specify the user login information.
`{`			
`password`	`1..1`	`base64Binary`	The password of the user account specified in the `auth_nameType` portion.
`}`			

Returns:

Name	Min/Max	Type	Description
`token`	`1..1`	`tokenType`	The security token for the user.

Returns error if the user credentials are invalid.

Description:

The call authenticates a user against the specified realm. Unlike the system/login call (p. 553), this call does not log the user in and does not create a session for the user. It simply verifies the identity of the user and, in case of success, returns the user security information.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <authenticate>
        <name>VGVzdF9Vc2Vy</name>
        <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        <password>bXlwYXNz</password>
      </authenticate>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="38c4ae0120ft66bbr2c8" time="2009-10-22T08:05:00+0000">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient60-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:authm>
        <ns2:token xsi:type="ns3:tokenType">
            <ns3:user>AQUAAAAAIAERFULeKciOQr59KBZnje7JAQAAAA==</ns3:user>
            <ns3:groups>
                <ns3:sid>AQUAAAAAIAERFULeKciOQr59KBZnje7JAQAAAA==</ns3:sid>
                <ns3:sid>AQUAAAAAIAGvZxwyACmNS7W3n7IpoA6KAQAAAA==</ns3:sid>
            </ns3:groups>
            <ns3:deny_only_sids/>
            <ns3:privileges/>
            <ns3:name xsi:type="ns3:auth_nameType">
                <ns3:name>VGVzdF9Vc2Vy</ns3:name>
                <ns3:realm>00000000-0000-0000-0000-000000000005</ns3:realm>
            </ns3:name>
        </ns2:token>
    </ns2:authm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

del_from_group

Summary:

Remove a user or a group from a group.

Request specification:

Name	Min/Max	Type	Description
`del_from_group`	`1..1`	`auth_nameType` (p. 29)	The member user or member group information.
`{`			
`group`	`1..[]`	`auth_nameType` (p. 29)	The list of the groups to remove the specified user/group from.
`}`			

Returns:

OK/Errors.

Description:

The call removes the specified user or group from the specified group(s). To delete a user or a group from the database (realm), use `del_user` (p. 89) or `del_group` (p. 87).

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <del_from_group>
        <name>VGVzdF9Vc2Vy</name>
        <realm>00000000-0000-0000-0000-000000000005</realm>
        <group>
          <name>Q049QWRtaW5pc3RyYXRvcnMsQ049Um9sZXMsREM9U1dB</name>
          <realm>00000000-0000-0000-0000-000000000005</realm>
        </group>
      </del_from_group>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="39c4ae01815t428br2c8" time="2009-10-22T08:30:42+0000">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient60-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:authm>
        <ns1:ok/>
    </ns2:authm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

del_group

Summary:

Deletes the specified group from the specified realm.

Request specification:

Name	Min/Max	Type	Description
`del_group`			

{			
group	0..1	auth_nameType (p. 29)	The group information.
}			

Returns:

OK/Errors

Description:

The del_group call deletes the specified group from the specified realm. If the group has member users and/or groups, their membership will be automatically revoked.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <del_group>
        <group>
          <name>Q049VGVzdF9Hcm91cCxDTj1Sb2xlcyxEQz1TV0E=</name>
          <realm>3f15a4a2-14e0-17da-b387-599bc36ee36d</realm>
        </group>
      </del_group>
    </authm>
  </data>
</packet>
```

Output

```
<packet id="2" version="4.0.0">
  <origin>authm</origin>
  <target>vzclient1</target>
  <dst>
    <director>gend</director>
  </dst>
  <data>
    <authm>
      <ok/>
    </authm>
  </data>
  <src>
    <director>opd</director>
  </src>
</packet>
```

del_realm

Summary:

Deletes an existing realm definition from the Agent configuration.

Request specification:

Name	Min/Max	Type	Description
del_realm			

{			
id	1..1	guid_type	The ID of the realm to remove.
}			

Returns:

OK/Errors.

Description:

The call deletes the specified realm definition from the Agent configuration. It does not physically delete the "database" (i.e. an LDAP directory instance), so the user data stored in it will not be affected. You can recreate the realm definition using the same directory at any time if you wish.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <del_realm>
        <id>453a0163-8abd-4e38-a04f-fe67d0c48b97</id>
      </del_realm>
    </authm>
  </data>
</packet>
```

del_user

Summary:

Permanently deletes the specified user(s) from their respective realms.

Call specification:

Name	Min/Max	Type	Description
del_user			
{			
user	1..[]	auth_nameType (p. 29)	The list of users to delete. You can delete multiple users from multiple realms at the same time -- simply specify the user name and the appropriate realm ID in each instance of the structure.
}			

Returns:

OK/Errors

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <del_user>
        <user>
          <name>VGVzdF9Vc2Vy</name>
          <realm>3f15a4a2-14e0-17da-b387-599bc36ee36d</realm>
        </user>
      </del_user>
    </authm>
  </data>
</packet>
```

edit_group

Summary:

Modifies an existing group.

Request specification:

Name	Min/Max	Type	Description
edit_group	1..1	auth_nameType (p. 29)	The auth_nameType portion of the request specification is used to specify the name/domain/realm information about the group that you would like to modify.
{			
group	1..1	sp_modType (p. 74)	The new group information. Includes the name and the group attributes. The attribute values can be added, deleted, or replaced based on the specified operation type. See sp_modType (p. 73) for details.
}			

Returns:

OK/Errors.

Example:

The following example modifies the `description` attribute value of the specified group.

Input

```
<packet version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <edit_group>
        <name>VGVzdF9Hcm91cA==</name>
        <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        <group>
          <data>
            <mod>
              <type>description</type>
              <value>VGhpcyBpcyBhIG5ldyBkZXNjcmlwdGlvbg==</value>
              <op>2</op>
            </mod>
          </data>
        </group>
      </edit_group>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="11c460a9dc5t2ea6r350" time="2007-03-21T13:10:28+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns1:ok/>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

edit_user

Summary:

Modifies an existing user.

Request specification:

Name	Min/Max	Type	Description
edit_user	1..1	auth_nameType (p. 29)	The auth_nameType portion of the request specification is used to specify the name/domain/realm information about the user that you would like to modify.
{			
user	1..1	sp_modType (p. 74)	The new user information. Includes the user name and the user attributes. The attribute values can be added, deleted, or replaced based on the specified operation type. See sp_modType (p. 73) for details.
password	0..1	base64Binary	New password.
}			

Returns:

OK/Error.

Example:

Modifying the existing user name and the value of the `description` attribute.

Input

```
<packet version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <edit_user>
        <name>VGVzdF9Vc2Vy</name>
        <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        <user>
          <name>Sm9obiBEb2U=</name>
          <data>
            <mod>
              <name>description</name>
              <value>Sm9obiBEb2UgdXNlcg==</value>
              <op>2</op>
            </mod>
          </data>
        </user>
      </edit_user>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="14c460a9eeft7e87r350" time="2007-03-21T13:13:36+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns1:ok/>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_auth_name

Summary:

Returns login information for the security account specified by its SID.

Request specification:

Name	Min/Max	Type	Description
get_auth_name			
{			
sid	1..1	sidType (p. 27)	Security ID (SID).
realm	0..1	guid_type (p. 26)	The ID of the realm in which you would like to search for the user.
}			

Returns:

Name	Min/Max	Type	Description
auth_name	1..1	auth_nameType (p. 29)	The security account login information.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <get_auth_name>
        <sid>AQQAAAAAAAXoAwAAQSVagwTHaUu6mzyE</sid>
        <realm>b5945e8a-68f9-48c2-827c-96af9549e46b</realm>
      </get_auth_name>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="39c461cde46t63cbr1d8" time="2007-04-11T13:10:39+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient13</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns2:auth_name xsi:type="ns3:auth_nameType">
      <ns3:name>QWRtaW5pc3RyYXRvcgA=</ns3:name>
      <ns3:realm>b5945e8a-68f9-48c2-827c-96af9549e46b</ns3:realm>
    </ns2:auth_name>
    <ns2:type>1</ns2:type>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_group

Summary:

Retrieves group information from the specified realm.

Request specification:

Name	Min/Max	Type	Description
`get_group`			
`{`			
`[`	`1..1`		Denotes a choice between the `group` and the `attr` elements. You have to include one of them, you cannot execute the call without any parameters.

group	1..1	auth_nameType (p. 29)	The group to retrieve the information for. The group is specified by supplying the group name, domain, and realm information. Omit the name parameter to retrieve all groups from the specified realm. If the name is included, only the information for the specified group will be retrieved.
attr	1..1	none	This parameter can be used to search for a group or multiple groups having a particular attribute set to a particular value. This applies only to the groups stored in an LDAP directory.
{			
name	1..1	string	Attribute name.
value	1..1	base64Binary	Attribute value.
realm	1..1	guid_type (p. 26)	The ID of the realm to conduct the search in.
}			
]			
attrs	0..1	none	The group attributes to include in the result set. Omit this element to retrieve just the names of a group (or groups) without any attributes. Omit the name element (below) to retrieve all available attributes.
{			
name	0..[]	string	The list of attributes to include in the result set. If a group doesn't have a particular attribute value set, the attribute will not be listed for that group.
}			
}			

Returns:

Name	Min/Max	Type	Description
group	1..[]	security_principalType (p. 74)	Group information.

Description:

The call retrieves the group information from the specified realm. The following actions are possible depending on the options selected:

- Retrieving the list of all groups from the specified realm. This is achieved by including the `group` element and specifying the realm ID.
- Retrieving the information for the specified group. This is achieved by including the `group` element and specifying the group name and the realm ID.
- Searching for a group or multiple groups based on the value of an attribute of a group. For this, include the `attr` element and specify the attribute name and value, and the realm ID to conduct the search in.

Note: If you are using an external LDAP directory, you have to make sure that the groups are stored as objects of class `Group` (`objectClass=Group`). If the group objects in your directory use a different class, the `get_group` call will not find them.

Example:

Retrieving the list of all groups from the specified realm. Requesting to include the value of the `description` attribute for every group in the result set.

Input

```
<packet version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <get_group>
        <group>
          <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        </group>
        <attrs>
          <name>description</name>
        </attrs>
      </get_group>
    </authm>
  </data>
</packet>
```

Output

```
<packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="14c469df342t26e9r4b4" time="2007-07-18T08:01:55+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient4-2cbbe469-8f57-7f46-97fb-fd987231d957</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns2:group>
      <ns2:data>
        <ns2:attr>
          <ns2:name>description</ns2:name>
          <ns2:value>VGhpcyBpcyBhIHRlc3QgZ3JvdXA=</ns2:value>
        </ns2:attr>
```

```
      </ns2:data>
      <ns3:name>VGVzdF9Hcm91cA==</ns3:name>
      <ns3:realm>3e761571-6607-1344-a064-a42679da8ed9</ns3:realm>
    </ns2:group>
    <ns2:group xsi:type="ns3:auth_nameType">
      <ns3:name>VmlydHVvenpvIENvbnRyb2wgQ2VudGVyIFVzZXJz</ns3:name>
      <ns3:realm>3e761571-6607-1344-a064-a42679da8ed9</ns3:realm>
    </ns2:group>
    <ns2:group>
      <ns2:data>
        <ns2:attr>
          <ns2:name>description</ns2:name>
          <ns2:value>VlpBZ2VudCBhZG1pbmlzdHJhdG9ycyBncm91cA==</ns2:value>
        </ns2:attr>
      </ns2:data>

<ns3:name>Y249dnphZG1pbmlzdHJhdG9ycyxjbj1CdWlsdGluLGRjPVZaTA==</ns3:name>
      <ns3:realm>3e761571-6607-1344-a064-a42679da8ed9</ns3:realm>
    </ns2:group>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</packet>
```

The next example demonstrates how to search for a group based on the value of a group attribute. Here, we are searching for a group with the `description` attribute set to "Agent administrators group".

Input

```
<packet version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <get_group>
        <attr>
          <name>description</name>
          <value>VlpBZ2VudCBhZG1pbmlzdHJhdG9ycyBncm91cA==</value>
          <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        </attr>
      </get_group>
    </authm>
  </data>
</packet>
```

Output

```
<packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="14c469df342t26e9r4b4" time="2007-07-18T08:01:55+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient4-2cbbe469-8f57-7f46-97fb-fd987231d957</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns2:group>

<ns3:name>Y249dnphZG1pbmlzdHJhdG9ycyxjbj1CdWlsdGluLGRjPVZaTA==</ns3:name>
      <ns3:realm>3e761571-6607-1344-a064-a42679da8ed9</ns3:realm>
    </ns2:group>
```

```
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</packet>
```

get_realm

Summary:

Retrieves the list of the available Realms from the current Agent configuration.

Request specification:

Name	Min/Max	Type	Description
get_realm	1..1		

Returns:

Name	Min/Max	Type	Description
realms			
{			
realm	1..[]	realmType (p. 54)	The list of the available realms. The appropriate data type will be used for a particular realm type (LDAP, System, etc.). See the subtypes of the realmType type (p. 54) for the available realm types.
}			

Description:

Use this call to retrieve the list of realms to present to the user during login or any other operation requiring the realm ID as an input parameter. To get the list of realms during the initial login to the system (p. 553), use the `system/get_realm` call (p. 545).

Example:

Input

```
<packet version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <get_realm/>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/dirm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="23c4adff204tbb3r2c8" time="2009-10-22T05:48:20+0000">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient60-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:authm>
        <ns2:realms>
            <ns2:realm xsi:type="ns3:realmType">
                <ns3:builtin/>
                <ns3:name>System</ns3:name>
                <ns3:type>0</ns3:type>
                <ns3:id>00000000-0000-0000-0000-000000000000</ns3:id>
            </ns2:realm>
            <ns2:realm xsi:type="ns3:realmType">
                <ns3:builtin/>
                <ns3:name>Parallels Internal</ns3:name>
                <ns3:type>4</ns3:type>
                <ns3:id>00000000-0000-0000-0000-000000000005</ns3:id>
            </ns2:realm>
            <ns2:realm xsi:type="ns3:realmType">
                <ns3:builtin/>
                <ns3:name>Parallels Agent Internal</ns3:name>
                <ns3:type>-1</ns3:type>
                <ns3:id>00000000-0000-0000-0000-000000000006</ns3:id>
            </ns2:realm>
            <ns2:realm xsi:type="ns4:dir_realmType">
                <ns4:login>
                    <ns4:domain>cWEuc3cucnU=</ns4:domain>
                    <ns4:name>YWRtaW5pc3RyYXRvcg==</ns4:name>
                    <ns4:realm>f3ddc2f0-0b5e-4220-b6a8-
a26616ff3548</ns4:realm>
                </ns4:login>
                <ns3:name>DataBase</ns3:name>
                <ns3:type>1</ns3:type>
                <ns3:id>f3ddc2f0-0b5e-4220-b6a8-a26616ff3548</ns3:id>
                <ns4:address>base.sw.ru</ns4:address>
```

```
                <ns4:port>389</ns4:port>
                <ns4:base_dn>sw.ru</ns4:base_dn>
                <ns4:default_dn>sw.ru</ns4:default_dn>
            </ns2:realm>
        </ns2:realms>
    </ns2:authm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

get_sid

Summary:

Returns SID of the security account specified by the login info.

Request specification:

Name	Min/Max	Type	Description
get_sid			
{			
auth_name	1..1	auth_nameType (p. 29)	The login information of the security account to search for.
}			

Returns:

Name	Min/Max	Type	Description
sid	1..1	sidType (p. 27)	Security ID.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>authm</target>
  <data>
    <authm>
      <get_sid>
        <auth_name>
          <name>VGVzdF9Vc2Vy</name>
          <realm>00000000-0000-0000-0000-000000000005</realm>
        </auth_name>
      </get_sid>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="17c460aa333t99r350"
time="2007-03-21T13:25:05+0000" priority="0" version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns2:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykBIAAA9QEAAA==</ns2:sid>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_user

Summary:

Retrieves user information from the specified realm.

Request specification:

Name	Min/Max	Type	Description
`get_user`	`1..1`		
`{`			
`[`	`1..1`		Denotes a choice between the `user` and the `attr` elements.

user	1..1	auth_nameType (p. 29)	The user to retrieve the information for. The user is specified by supplying the user name, domain, and realm information. Omit the name parameter to retrieve all users from the specified realm. If the name is included, only the information for the specified user will be retrieved.
attr	1..1		This parameter can be used to search for a user or multiple users having a particular attribute set to a particular value. This applies only to the users stored in an LDAP directory.
{			
name	1..1	string	The name of the attribute to search for.
value	1..1	base64Binary	Attribute value.
realm	1..1	guid_type (p. 26)	The ID of the realm to conduct the search in.
}			
]			
attrs	0..1	none	The user attributes to include in the result set. Omit this element to retrieve just the names of a user (or users) without any attributes. Omit the name element (below) to retrieve all available attributes.
{			
name	0..[]	string	The list of attributes to include in the result set. If a user doesn't have a particular attribute value set, the attribute will not be listed for that user.
}			
}			

Returns:

Name	Min/Max	Type	Description
user	1..[]	security_principalType (p. 74)	User information.

Description:

The call retrieves the user information from the specified realm. The following actions are possible depending on the options selected:

- Retrieving the list of all users from the specified realm. This is achieved by including the `user` element and specifying the realm ID.
- Retrieving the information for the specified user. This is achieved by including the `user` element and specifying the user name and the realm ID.
- Searching for a user or multiple users based on the value of an attribute of a user. For this, include the `attr` element and specify the attribute name and value, and the realm ID to conduct the search in.

Note: If you are using an external LDAP directory, you have to make sure that the users are stored as objects of class `User` (`objectClass=User`). If the user objects in your directory use a different class, the `get_user` call will not find them.

Example:

Retrieving the list of all users from the specified realm. Requesting to include the value of the `description` attribute for every user in the result set.

Input

```
<packet version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <get_user>
        <user>
          <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        </user>
        <attrs>
          <name>description</name>
        </attrs>
      </get_user>
    </authm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="24c469e07f4t1547r4b4" time="2007-07-18T08:59:11+0000" priority="0"
version="4.0.0">
<ns1:origin>authm</ns1:origin>
<ns1:target>vzclient4-2cbbe469-8f57-7f46-97fb-fd987231d957</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:authm>
    <ns2:user>
      <ns2:data>
        <ns2:attr>
          <ns2:name>description</ns2:name>
          <ns2:value>VGhpcyBpcyBhIHRlc3QgdXNlcg==</ns2:value>
```

```
        </ns2:attr>
      </ns2:data>
      <ns3:name>VGVzdF9Vc2Vy</ns3:name>
      <ns3:realm>3e761571-6607-1344-a064-a42679da8ed9</ns3:realm>
    </ns2:user>
  </ns2:authm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</packet>
```

set_realm

Summary:

Modifies the specified realm definition.

Request specification:

Name	Min/Max	Type	Description
`set_realm`			
`{`			
`realm`	`1..1`	`realmType` (p. 54)	Realm information. Depending on the realm type, use the appropriate subtype of the `realmType` type (p. 54). The `realm/id` parameter is mandatory and must contain the ID of the realm that you would like to modify.
`password`	`0..1`	`base64Binary`	The password for the user account specified in the `login` portion of the `realm` structure above.
`}`			

Returns:

OK/Error.

Description:

The `set_realm` call allows to modify parameters that define an existing realm. You can change realm name, connection parameters (address, port, etc.), login parameters, and password. For more information on individual realm parameters see the `add_realm` call (p. 78).

Example:

Input

```
<packet xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>authm</target>
  <data>
    <authm>
      <set_realm>
        <realm xsi:type="ns4:dir_realmType">
           <id>18e8c5f0-e656-4144-864c-0520275a4bd1</id>
           <type>1</type>
           <name>myrealm</name>
           <address>192.168.0.117</address>
           <port>398</port>
           <base_dn>dc=vzl</base_dn>
           <default_dn>cn=users,dc=vzl</default_dn>
           <login>
             <name>Y249dnphZ2VudCxkYz1WWkw=</name>
           </login>
         </realm>
         <password>bXlwYXNz</password>
      </set_realm>
    </authm>
  </data>
</packet>
```

backup_storagem

Purpose:

The `backup_storagem` interface provides calls for configuring a backup storage.

Specification:

The interface is derived from the `data_storagem` interface (p. 157).

Calls

Call	Description
get_storage_config (p. 159)	Retrieves backup storage configuration.
set_storage_config (p. 159)	Sets backup storage configuration.

backupm

Purpose:

The backup management interface.

Types

backup_dataType

Summary:

Basic backup information. Used in `get_info` call (p. 136).

Type specification:

The type has no elements.

Subtypes:

`env_backup_dataType` (p. 111)

backup_options_baseType

Summary:

Base type for backup options.

Type specification:

The type has no elements.

Subtypes:

`backup_optionsType` (p. 107)

backup_optionsType

Summary:

Backup options.

Type specification:

Extends `backup_options_baseType (p. 107)`

Adds the following elements:

Name	Min/Max	Type	Description

type	0..1	int	Backup type: 0 -- Full (default). A full backup is a starting point for all other backup types. You may define the files and folders to be backed up using the include/exclude options (see below). 1 -- Incremental. Only the files that have changed since the latest full, incremental, or differential backup are included. When restoring from an incremental backup, you'll need the latest full backup as well as every incremental and/or differential backup that you've made since the last full backup. 2 -- Differential. Only the files that have changed since the last full backup are included. When restoring from a differential backup, only the latest differential backup itself and the latest full backup is needed.
policy	0..1		Backup policy.
{			
ignore_error	0..1	none	A flag indicating not to stop on error when backing up multiple servers. If an individual server backup fails, the operation will proceed with the next server in the list.
ignore_unexistent	0..1	none	A flag indicating not to produce an error if one of the servers in the list does not exist. If this element is absent and the list contains an invalid Server ID, the batch backup operation will stop with an error.
}			
remove	0..1		If present, defines the old backup of the specified server to be removed from the backup server.
{			

backup	0..1	string	Backup ID to remove. If this element is absent, the oldest backup from this sequence will be removed.
}			
include_list	0..1		Files and directories to include in the backup. Use this option when you want to backup only the select files and directories.
{			
path	1..[]	base64Binary	A list of files and directories to include in the backup. Wildcards are permitted.
}			
exclude_list	0..1	exclude_list Type (p. 113)	Files and directories to exclude from the backup. The list provides various options that can be used individually or in any combination.
compression	0..1	int	Compression level: 0 -- no compression 1 -- normal (default) 2 -- high 3 -- maximum
description	0..1	base64Binary	Backup description.

backupid_type

Summary:

Backup ID.

Type specification:

Restriction: ds_object_path_type (p. 158)

backupm_configType

Summary:

Contains the backup configuration information.

Type specification:

Name	Min/Max	Type	Description
backup_server	1..1	connection_info_baseType	Backup server connection information.
chain_length	0..1	int	Maximum number of incremental backups in any given backup sequence. When this number is exceeded, a new backup sequence must be started beginning with a full backup.
chain_days	0..1	int	Maximum number of days an incremental backup sequence may continue uninterrupted. When this number is exceeded, a new sequence must be started beginning with a new backup.
keep_max	0..1	int	The maximum number of backup archives for a given server that should be kept on the backup server. When this number is exceeded, the oldest backup file is automatically removed on every successful new backup.
compression	0..1	int	The default compression level. See backup_optionsType (p. 107) for the available compression levels.
type	0..1	int	Default backup type. See backup_optionsType (p. 107) for the list of backup types.
pe_backups_limit	0..1	none	The maximum number of backups a user can perform on a single server.

backupType

Summary:

Backup information.

Type specification:

Extends `ds_object_infoType (p. 158)`

Adds the following elements:

Name	Min/Max	Type	Description
`eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`description`	`0..1`	`base64Binary`	Backup description.
`count`	`1..1`	`int`	The total number of backups for this server in this storage.
`capability`	`0..1`		Backup capabilities. Specifies miscellaneous backup properties.
`{`			
`browsable`	`0..1`	`none`	A flag indicating that the contents of this backup can be listed using the `filer/list` call (p. 239), and can be restored using the `selective_restore_env` call (p. 125).
`}`			

env_backup_dataType

Summary:

Contains a server-specific backup information. Used in `get_info` call (p. 136).

Type specification:

Extends `backup_dataType (p. 107)`

Adds the following elements:

Name	Min/Max	Type	Description
`env`	`0..1`	`envType (p. 35)`	The server information.

include_list	0..1		A list of files and directories that were listed in the include_list element in the original backup request (p. 107).
{			
path	1..[]	base64Binary	Pathnames (may contain wildcards).
}			
exclude_list	0..1	exclude_listType (p. 113)	A list of files and directories that were listed in the exclude_list element in the original backup request (p. 107).

get_env_info_optionsType

Summary:

Server-specific options for get_info call (p. 136).

Type specification:

Extends get_info_optionsType (p. 113)

Adds the following elements:

Name	Min/Max	Type	Description
env	0..1	none	If this element is present, the get_info call returns the full server information.
excludes	0..1	none	If present, the get_info call returns the names of files and directories that were included in or excluded from the backup. See exclude_list and include_list in backup_optionsType (p. 107) for more info.

exclude_listType

Summary:

Contains files and directories to exclude from backup.

Type specification:

Name	Min/Max	Type	Description
`path`	`0..[]`	`base64Binary`	List of files and directories to exclude from backup.
`hidden`	`0..[]`	`none`	If this element is present, all hidden files will be excluded from the backup.
`system`	`0..1`	`none`	If this element is present, all system files will be excluded from the backup. This option may not be available for all server types. You can check whether this option is available or not for a particular server type by searching the Agent vocabulary for the `backup.exclude.system` entry. If the entry is present, then the feature is available. To retrieve the vocabulary, use the `get_vocabulary` call (p. 551).

get_info_optionsType

Summary:

Basic options for `get_info` call (p. 136).

Type specification:

This is an abstract type. Use the appropriate subtype of this type when executing the `get_info` call (p. 136).

Subtypes:

`get_env_info_optionsType` (p. 112)

list_optionsType

Summary:

Backup list retrieval options.

Type specification:

Name	Min/Max	Type	Description
`eid`	`0..1`	`eid_type` (p. 25)	Server ID to list the backups for.
`latest`	`0..1`	`none`	If this element is present, only the latest backup for each existing Container (or the Container specified in the `eid` element above) will be included in the result. If this element is omitted, the list will contain all of the available backups -- old and new.
`info`	`0..1`	`none`	If this element is present, the additional backup information will be retrieved, if available. The information will be included in the `info` element of the `backup` return structure (p. 129).
`storage_eid`	`0..1`	`eid_type` (p. 25)	The Server ID of the slave Node in the Virtuozzo group from which to retrieve the list of backups. The parameter can be used only when the `list` call (p. 129) is executed on the Master Node in the group. When retrieving a list of backups from a standalone Hardware Node, this parameter has no effect.

remove_optionsType

Summary:

Backup removal options.

Type specification:

Name	Min/Max	Type	Description
`eid`	`0..1`	`eid_type` (p. 25)	Server ID. If this element is included, the `remove` call (p. 132) will search for all backups of the specified server and will remove all of them.
`prev`	`0..1`	`none`	If this element is included, and the `backup_id` element of the `remove` call (p. 132) is included as well, the call will find and remove all prior backups of the same server that the specified `backup_id` belongs to. In all other cases, the element is ignored.

restore_optionsType

Summary:

Backup restoration options.

Type specification:

Name	Min/Max	Type	Description
`force`	`0..1`	`none`	Force the restore operation. If the element is present, the procedure will try to resolve potential conflicts, or ignore them if no resolution is possible. The conflicts may arise due to a duplicate server name or a Virtuozzo Container ID (veid).

search_optionsType

Summary:

Backup search options.

Type specification:

Name	Min/Max	Type	Description
`hostname`	`0..1`	`string`	Server hostname (wildcards supported).

ip	0..1	string	Server IP address. (wildcards supported).
start_date	0..1	datetimeType	Search with specified backup creation date range.
end_date	0..1	datetimeType	Search with specified backup creation date range.

selective_restore_optionsType

Summary:

Selective restore options.

Type specification:

Name	Min/Max	Type	Description
skip_locked	0..1	none	If this element is included, the restore operation will not stop on error while trying to restore a locked file. If the element is omitted, an attempt to restore a locked file will produce an error and the entire operation will be canceled.

Calls

Call	Description
backup_env (p. 117)	Backs up a server.
restore_env (p. 121)	Restores a server.
selective_restore_env (p. 125)	Selectively restores individual files from a backup.
list (p. 129)	Retrieves backup information.
remove (p. 132)	Removes a backup.
search (p. 134)	Searches for a server backup.
get_info (p. 136)	Gets extended information about a backup.
set_config (p. 137)	Sets the backup manager configuration.
get_config (p. 139)	Retrieves the backup manager configuration information.

backup_env

Summary:

Backs up a server. You may specify multiple servers to back up at the same time.

Request specification:

Name	Min/Max	Type	Description
backup_env			
{			
env_list	1..1	eid_listType (p. 32)	Servers to backup.
backup_options	0..1	backup_options_baseType (p. 107)	Backup options.
backup_server	0..1	connection_infoType (p. 30)	Backup server connectivity information for remote backups. If this element is omitted, the default backup server configuration will be used. To retrieve the default configuration, use the get_configuration (p. 533) call.
}			

Returns:

Name	Min/Max	Type	Description
backup	0..[]	backupType (p. 111)	Backup details. When backing up multiple servers, a mix of the backup and error information may be returned.

Description:

The call provides a set of options that allow you to control the backup operation. The options include replacing a specific old backup archive, backing up only the directories that you need, excluding the files and directories that you don't need, setting the compression level, and others. See `backup_optionsType` (p. 107) for the complete list of options.

Since a backup operation can take a significant amount of time, you may optionally include the `progress="on"` attribute and specify the packet ID to receive the progress information. The progress information will be sent to your client program via a series of responses using the `progress` packet (p. 563).

Example:

Performing a full backup of the specified Virtuozzo Container. Setting a compression level to "high". Backing up to the default defined backup server.

Input

```
<packet progress="on" log="on" id="2" version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <backup_env>
        <env_list>
          <eid>57c2cd6c-c02b-4645-bdb5-e883ea005896</eid>
        </env_list>
        <backup_options>
          <type>0</type>
          <compression>2</compression>
          <description>RnVsbCBiYWNrdXAgMjAwNy0wMS0xMg==</description>
        </backup_options>
      </backup_env>
    </backupm>
  </data>
</packet>
```

Progress Messages

The following are some of the progress messages that we received in this example (the actual XML packets are not listed here for brevity):

```
Operation backup_env is started
Checking parameters
Dumping quota
Backup storage: preparing to backup
Adjusting backup type (full)
Backup storage: receiving backup file
Backing up private area.

// Some progress percent messages were received here...
Percent: 2
...
Percent: 54
...
Percent: 99
....

Backup storage: storing private backup data
Backup storage: filling resultant backup info
```

The following is the actual example of a packet containing a progress message.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="2" time="2007-02-
16T16:14:04+0000" type="1" priority="4000" version="4.0.0">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns1:progress>
    <ns1:op>backupm.backup_env</ns1:op>
    <ns1:message>
      <ns1:message>T3BlcmF0aW9uICVvcF9uYW1lJSBpcyAlc3RhdHVzJQ==</ns1:message>
      <ns1:name></ns1:name>
      <ns1:translate/>
      <ns1:parameter>
        <ns1:message>YmFja3VwX2Vudg==</ns1:message>
        <ns1:name>op_name</ns1:name>
      </ns1:parameter>
      <ns1:parameter>
        <ns1:message>c3RhcnRlZA==</ns1:message>
        <ns1:name>status</ns1:name>
        <ns1:translate/>
      </ns1:parameter>
    </ns1:message>
    <ns1:status>1</ns1:status>
  </ns1:progress>
</ns1:data>
<ns1:target>log_subscription</ns1:target>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Output

The following is a packet received on the backup operation completion. The packet contains the backup information, including the information about the server that was backed up, the backup ID, backup archive size, backup type, and other info.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/data_storagem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" id="b6c4ae06248t2c49r2c8" time="2009-10-22T13:47:34+0000">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient67-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:backupm>
        <ns2:backup>
            <ns2:eid>87af640a-f151-2844-863b-c7f4153d79b4</ns2:eid>
            <ns2:description>aXZyaWJhY2t1cA==</ns2:description>
            <ns2:count>2</ns2:count>
            <ns2:capability>
                <ns2:browsable/>
            </ns2:capability>
            <ns3:id>87af640a-f151-2844-863b-
c7f4153d79b4/20091022134653</ns3:id>
            <ns3:time>2009-10-22T13:46:53+0000</ns3:time>
```

```
            <ns3:size>4394684</ns3:size>
            <ns3:type>0</ns3:type>
            <ns3:storage_eid>e7beabe4-89c5-bc42-b45a-
ca85449b3eb5</ns3:storage_eid>
            <ns3:info xsi:type="ns4:infoType">
                <ns4:message></ns4:message>
                <ns4:name></ns4:name>
                <ns4:translate/>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
                    <ns4:name>hostname</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>MTAuMzEuMzIuMTA=</ns4:message>
                    <ns4:name>ip</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
                    <ns4:name>name</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message></ns4:message>
                    <ns4:name>os</ns4:name>
                    <ns4:parameter>
                        <ns4:message>TGludXg=</ns4:message>
                        <ns4:name>platform</ns4:name>
                    </ns4:parameter>
                </ns4:parameter>
                <ns4:parameter>

<ns4:message>ZTdiZWFiZTQtODljNS1iYzQyLWI0NWEtY2E4NTQ0OWIzZWI1</ns4:message>
                    <ns4:name>parent_eid</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>bWNjcDI1LnFhLnN3LnJ1</ns4:message>
                    <ns4:name>parent_env_title</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>bWNjcDI1LnFhLnN3LnJ1</ns4:message>
                    <ns4:name>storage_title</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Z2VuZXJpYw==</ns4:message>
                    <ns4:name>storage_type</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
                    <ns4:name>title</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>dmlydHVvenpv</ns4:message>
                    <ns4:name>type</ns4:name>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Njg5MzUyNQ==</ns4:message>
                    <ns4:name>veid</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
            </ns3:info>
```

```
        </ns2:backup>
    </ns2:backupm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

restore_env

Summary:

Restores a server from a backup.

Request specification:

Name	Min/Max	Type	Description
restore_env			
{			
backup_id	1..1	backupid_type (p. 109)	Backup ID. The ID is generated and returned to the client program at the end of the backup operation (p. 117). To get the list of the existing backups, use the list call (p. 129). To search for backups of a particular server, use the search call (p. 134).
restore_options	0..1	restore_optionsType (p. 115)	Restore options.
backup_server	1..1	connection_infoType (p. 30)	Backup server connectivity information for remote backups. If this element is omitted, the default backup server configuration will be used. To retrieve the default configuration, use the get_configuration (p. 533) call.

parent_eid	0..1	eid_type (p. 25)	This parameter is used only when restoring a Container within a Virtuozzo group. Normally, in a Virtuozzo group, a Container is restored to the original Hardware Node. If the original Hardware Node is not found, the restore operation will fail. In such a situation, you can use this parameter to specify the alternate Hardware Node to which to restore the Container. The following conditions apply: 1. The call must be executed on the Master Node in the group. 2. If the original Hardware Node exists, this parameter will be ignored.
}			

Returns:

OK/Error

Description:

Use the `restore_env` call to restore a server from a backup. The backup that you are restoring from must be one of the following:

- A full backup containing all the files and directories that are required for the server to operate properly.
- An incremental backup, plus all the prior incremental and differential backups, and the original full backup from the same sequence.
- A differential backup, plus the original full backup from the same sequence.

By default, a server will be restored to the host that you are currently connected to. In a Virtuozzo group, a server will be restored to the original Node. If the original Node is no longer registered with the group, use the `parent_eid` parameter to specify an alternate Node.

You may set the `progress="on"` and the `id` arguments in the packet element of the message header if you would like to receive the progress reports during the restore operations.

Example:

Restoring a server from the specified backup located on the default backup server. The server will be restored to the host that we are currently connected to.

Input

```
<packet progress="on" log="on" id="2" version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <restore_env>
        <backup_id>57c2cd6c-c02b-4645-bdb5-
e883ea005896/20070219150134</backup_id>
        <restore_options>
          <force/>
        </restore_options>
      </restore_env>
    </backupm>
  </data>
</packet>
```

Progress messages

The following is an example of one of the progress reports. When decoded, the message reads:

```
Operation restore_env started.
```

See `backup_env` (p. 117) and `progress` packet (p. 563) for more info on progress messages.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/progress_event"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" type="1" time="2009-10-22T14:53:40+0000"
id="b8c4ae071det2fffr2c8">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient67-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <progress>
        <ns2:op>backupm.restore_env</ns2:op>
        <ns2:message xsi:type="ns3:infoType">

<ns3:message>T3BlcmF0aW9uIHdpdGggdGhlIENvbnRhaW5lciAlZW52JSBpcyAlc3RhdHVzJQ==<
/ns3:message>
            <ns3:name></ns3:name>
            <ns3:translate/>
            <ns3:parameter>
                <ns3:message>JXRpdGxlJQ==</ns3:message>
                <ns3:name>env</ns3:name>
                <ns3:translate/>
                <ns3:parameter>

<ns3:message>ODdhZjY0MGEtZjE1MS0yODQ0LTg2M2ItYzdmNDE1M2Q3OWI0</ns3:message>
                    <ns3:name>eid</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>Q1QzbGluUFNCTQ==</ns3:message>
                    <ns3:name>title</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>dmlydHVvenpv</ns3:message>
                    <ns3:name>type</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
            </ns3:parameter>
            <ns3:parameter>
                <ns3:message>YmFja3VwbS5yZXN0b3JlX2Vudg==</ns3:message>
                <ns3:name>op_name</ns3:name>
            </ns3:parameter>
            <ns3:parameter>
                <ns3:message>JXRpdGxlJQ==</ns3:message>
                <ns3:name>source_env</ns3:name>
                <ns3:translate/>
                <ns3:parameter>

<ns3:message>MmZhYmUxOTQtMDQ3Ny0wODcxLWEzMmQtNDkyMjAwNTBjZjVj</ns3:message>
                    <ns3:name>eid</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>bWNjcDQx</ns3:message>
                    <ns3:name>title</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
                <ns3:parameter>
                    <ns3:message>Z2VuZXJpYw==</ns3:message>
                    <ns3:name>type</ns3:name>
                    <ns3:translate/>
                </ns3:parameter>
            </ns3:parameter>
```

```
            <ns3:parameter>
                <ns3:message>c3RhcnRlZA==</ns3:message>
                <ns3:name>status</ns3:name>
                <ns3:translate/>
            </ns3:parameter>
        </ns2:message>
        <ns2:status>1</ns2:status>
        <ns2:eid_list>
            <ns3:eid>87af640a-f151-2844-863b-c7f4153d79b4</ns3:eid>
        </ns2:eid_list>
    </progress>
</ns1:data>
<target>log_subscription</target>
<src>
    <director>gend</director>
</src>
</packet>
```

Output

The following "OK" message is received on successful restoration.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="bc45dc4ddbt6df1r488"
time="2009-10-22T14:54:04+0000" priority="4000" version="4.0.0">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:backupm>
    <ns1:ok/>
  </ns2:backupm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

selective_restore_env

Summary:

Selectively restore files from a backup.

Request specification:

Name	Min/Max	Type	Description
selective_restore_env			
{			
eid	1..1	eid_type (p. 25)	The ID of the server to restore the selected files to.

backup_id	1..1	backupid_type (p. 109)	Backup ID. The ID is generated and returned to the client program at the end of the backup operation (p. 117). To get the list of the existing backups, use the list call (p. 129). To search for backups for a particular server, use the search call (p. 134).
path	1..[]	base64Binary	A list of files and directories to restore. Wildcards are permitted.
selective_restore_options	0..1	selective_restore_optionsType (p. 116)	Restore options.
backup_server	0..1	connection_infoType (p. 30)	Backup server connectivity information for remote backups. If this element is omitted, the default backup server configuration will be used. To retrieve the default configuration, use the get_configuration (p. 533) call.
}			

Returns:

OK/Error

Description:

Use the `selective_restore_env` call to restore only the files and directories that you need or to restore from a backup that does not contain all the files and directories that are required for the server to operate properly. There are some important differences between the regular `restore_env` call (p. 121) and this call.

First of all, since you can only restore the files into an existing server, you must specify its Server ID. If the server doesn't exist, or cannot be found, the call will fail. Please note that the call will try to locate the specified server on the current host. Even if you execute the call on the Master Node in a Virtuozzo group, it will still look for the specified target server on the Master Node only. If your server is hosted by another node in a group, you will have to connect to it directly or provide its ID via the `dst` element in the message header.

You always have to specify the files and directories that you would like to restore. In addition, you may restore the files and directories into any available server of the same type as the original server.

As with other backup and restore operations, you may set the `progress="on"` and the `id` arguments in the packet element of the message header if you would like to receive progress reports during the restore operations.

Note: You can use the selective restore operation only with the backups that are capable of it. Use the `list` call (p. 129) to get the backup information and look for the `capabilities/browsable` flag (p. 111) in the result set. If the element is present, the selective restore operation can be performed on the backup.

Example:

Restoring the `/var/tmp` directory from the specified backup into the specified Virtuozzo Container.

Input

```
<packet progress="on" log="on" id="2" version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <selective_restore_env>
        <eid>57c2cd6c-c02b-4645-bdb5-e883ea005896</eid>
        <backup_id>57c2cd6c-c02b-4645-bdb5-
e883ea005896/20070219150134</backup_id>
        <path>L3Zhci90bXA=</path>
      </selective_restore_env>
    </backupm>
  </data>
</packet>
```

Progress messages:

The following is an example of one of the progress reports (this was the last progress message received, actually). When decoded, the message reads as follows:

```
Operation selective_restore_env finished successfully.
```

See backup_env (p. 117) and progress packet (p. 563) for more info on progress messages.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="cc45dc6bdbt5af1r488"
time="2007-02-20T09:00:16+0000" type="1" priority="4000" version="4.0.0">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:progress>
    <ns1:op>backupm.selective_restore_env</ns1:op>
    <ns1:message>

<ns1:message>T3BlcmF0aW9uICVvcF9uYW1lJSBpcyAlc3RhdHVzJSBzdWNjZXNzZnVsbHku</ns1
:message>
      <ns1:name></ns1:name>
      <ns1:translate/>
      <ns1:parameter>
        <ns1:message>c2VsZWN0aXZlX3Jlc3RvcmVfZW52</ns1:message>
        <ns1:name>op_name</ns1:name>
      </ns1:parameter>
      <ns1:parameter>
        <ns1:message>ZmluaXNoZWQ=</ns1:message>
        <ns1:name>status</ns1:name>
        <ns1:translate/>
      </ns1:parameter>
    </ns1:message>
    <ns1:percent>100</ns1:percent>
    <ns1:status>3</ns1:status>
  </ns1:progress>
</ns1:data>
<ns1:target>log_subscription</ns1:target>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Output

The final "OK" response on operation completion.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="cc45dc6bdbt5af1r488"
time="2007-02-20T09:00:16+0000" priority="4000" version="4.0.0">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:backupm>
    <ns1:ok/>
  </ns2:backupm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

list

Summary:

Retrieves backup information.

Request specification:

Name	Min/Max	Type	Description
`list`			
`{`			
`backup_id`	`0..[]`	`backupid_type` (p. 109)	List of backup IDs to retrive the information for. If this element is omitted, the information about all available backups will be retrieved.
`options`	`0..1`	`list_optionsType` (p. 114)	List options. Use the options provided by this element as a filter to retrieve the information that you require.
`}`			

Returns:

Name	Min/Max	Type	Description
`backup`	`0..[]`	`backupType` (p. 111)	Backup information.

Description:

If you are connected to the master node in a Virtuozzo group, the call retrieves information about all backups in the group. In all other cases, the call retrieves only the backups that are stored on the machine that you are connected to. If you are connected to a slave node but want to retrieve the information from a backup server located elsewhere, you must connect to it directly and execute the call there.

Example:

Retrieving information about the latest backup of the specified Virtuozzo Container.

Input

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <list>
        <options>
          <eid>57c2cd6c-c02b-4645-bdb5-e883ea005896</eid>
          <latest/>
        </options>
      </list>
    </backupm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/data_storagem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="6ac4ae03d97t3bf6r2c8" time="2009-10-22T11:10:41+0000">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient65-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:backupm>
        <ns2:backup>
            <ns2:eid>87af640a-f151-2844-863b-c7f4153d79b4</ns2:eid>
            <ns2:description>M3JkIHJlbW90ZSBvbiB3Mms4</ns2:description>
            <ns2:count>3</ns2:count>
            <ns2:capability>
                <ns2:browsable/>
            </ns2:capability>
            <ns3:id>87af640a-f151-2844-863b-
c7f4153d79b4/20091020094521</ns3:id>
            <ns3:time>2009-10-20T09:45:21+0000</ns3:time>
            <ns3:size>4386538</ns3:size>
            <ns3:type>0</ns3:type>
            <ns3:storage_eid>a6f1d061-8bcd-8ec7-1a73-
b078fe2d416f</ns3:storage_eid>
            <ns3:info xsi:type="ns4:infoType">
                <ns4:message></ns4:message>
                <ns4:name></ns4:name>
                <ns4:translate/>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
```

```
                    <ns4:name>hostname</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>MTAuMzEuMzIuMTA=</ns4:message>
                    <ns4:name>ip</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
                    <ns4:name>name</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message></ns4:message>
                    <ns4:name>os</ns4:name>
                    <ns4:parameter>
                        <ns4:message>TGludXg=</ns4:message>
                        <ns4:name>platform</ns4:name>
                    </ns4:parameter>
                </ns4:parameter>
                <ns4:parameter>

<ns4:message>ZTdiZWFiZTQtODljNS1iYzQyLWI0NWEtY2E4NTQ0OWIzZWI1</ns4:message>
                    <ns4:name>parent_eid</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>bWNjcDI1LnFhLnN3LnJ1</ns4:message>
                    <ns4:name>parent_env_title</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>bWNjcDQ0</ns4:message>
                    <ns4:name>storage_title</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Z2VuZXJpYw==</ns4:message>
                    <ns4:name>storage_type</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
                    <ns4:name>title</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>dmlydHVvenpv</ns4:message>
                    <ns4:name>type</ns4:name>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Njg5MzUyNQ==</ns4:message>
                    <ns4:name>veid</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
            </ns3:info>
        </ns2:backup>
    </ns2:backupm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

remove

Summary:

Removes a backup archive.

Request specification:

Name	Min/Max	Type	Description
`remove`			
`{`			
`backup_id`	`0..1`	`backupid_type (p. 109)`	The ID of the backup to be removed.
`options`	`0..1`	`remove_optionsType (p. 115)`	Removal options.
`}`			

Returns:

OK/Error

Description:

If you are connected to the master node in a Virtuozzo group, the call can remove any backup stored on any physical node in a cluster. In all other cases, the call can remove only the backups that are stored locally. If you are connected to a slave node but want to remove a backup from a machine located elsewhere, you must connect to it directly and execute the call there.

To remove a specific backup, supply the backup ID using the `backup_id` element. If, at the same time, you'll also include the `options/prev` element, the call will automatically determine the server that the specified backup belongs to, and will remove all prior backups that belong to the same server.

To remove all of the backups for a specific server, specify its Server ID using the `options/eid` element.

Example 1:

Removing the specified backup archive.

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <remove>
        <backup_id>57c2cd6c-c02b-4645-bdb5-
e883ea005896/20070219170146</backup_id>
      </remove>
    </backupm>
  </data>
</packet>
```

Example 2:

Removing a backup archive and all the prior backups belonging to the same server.

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <remove>
        <backup_id>57c2cd6c-c02b-4645-bdb5-
e883ea005896/20070219170146</backup_id>
         <options>
           <prev/>
         </options>
      </remove>
    </backupm>
  </data>
</packet>
```

Example 3:

Removing all backups belonging to the specified server.

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
```

```
    <backupm>
      <remove>
        <options>
         <eid>57c2cd6c-c02b-4645-bdb5-e883ea005896</eid>
        </options>
      </remove>
    </backupm>
  </data>
</packet>
```

search

Summary:

Searches existing backup archives for a specific server.

Request specification:

Name	Min/Max	Type	Description
search			
{			
options	1..1	search_optionsType (p. 107)	Search options. You may specify any option or any combination of the options.
}			

Returns:

Backup information if the specified server was found or an empty structure otherwise.

Name	Min/Max	Type	Description
backup	0..[]	backupType (p. 111)	Backup information.

Description:

If you are connected to the master node in a Virtuozzo group, the call will search through all backups located on every physical node. If you are connected to a slave node, the call will perform the search locally. To search for a backup on a backup server located elsewhere, connect to that server and execute the call there.

Example:

Searching backups by the hostname of the server.

Input

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <search>
        <options>
          <hostname>Host-106</hostname>
        </options>
      </search>
    </backupm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/data_storagem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="77c4ae046e6t366br2c8" time="2009-10-22T11:50:23+0000">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient65-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:backupm>
        <ns2:backup>
            <ns2:eid>87af640a-f151-2844-863b-c7f4153d79b4</ns2:eid>
            <ns2:description>M3JkIHJlbW90ZSBvbiB3Mms4</ns2:description>
            <ns2:count>3</ns2:count>
            <ns2:capability>
                <ns2:browsable/>
            </ns2:capability>
            <ns3:id>87af640a-f151-2844-863b-
c7f4153d79b4/20091020094521</ns3:id>
            <ns3:time>2009-10-20T09:45:21+0000</ns3:time>
            <ns3:size>4386538</ns3:size>
            <ns3:type>0</ns3:type>
            <ns3:storage_eid>a6f1d061-8bcd-8ec7-1a73-
b078fe2d416f</ns3:storage_eid>
            <ns3:info xsi:type="ns4:infoType">
                <ns4:message></ns4:message>
                <ns4:name></ns4:name>
                <ns4:translate/>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluUFNCTQ==</ns4:message>
                    <ns4:name>hostname</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
```

```
                <ns4:parameter>
                    <ns4:message>MTAuMzEuMzIuMTA=</ns4:message>
                    <ns4:name>ip</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
                <ns4:parameter>
                    <ns4:message>Q1QzbGluFNCTQ==</ns4:message>
                    <ns4:name>name</ns4:name>
                    <ns4:translate/>
                </ns4:parameter>
     <!-- The rest of the output is omitted for brevity -->
            </ns3:info>
         </ns2:backup>
    </ns2:backupm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

get_info

Summary:

Retrieves extended information about backup.

Request specification:

Name	Min/Max	Type	Description
get_info	1..1		
{			
backup_id	1..1	backupid_type (p. 109)	The backup ID.
options	1..1	get_info_optionsType (p. 113)	Options.
}			

Returns:

Name	Min/Max	Type	Description
info	1..1	backup_dataType (p. 107)	The requested backup information. The data type of the returned structure is determined by the data type of the options element in the request.

Description:

If you are connected to the Master Node in a Virtuozzo group, the call can access any backup located on every physical node. If you are connected to a slave node, the call can access only the local backups. If your backups are stored on a remote backup server, you must connect to that server directly and execute the call there.

Example:

Retrieving information about the specified backup.

```
<packet  xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <get_info>
        <backup_id>57c2cd6c-c02b-4645-bdb5-
e883ea005896/20070219170146</backup_id>
        <options xsi:type="ns2:get_env_info_optionsType">
          <env/>
        </options>
       </get_info>
   </backupm>
 </data>
</packet>
```

set_config

Summary:

Sets the default backup configuration.

Request specification:

Name	Min/Max	Type	Description
set_config			
{			
backupm_config	1..1	backupm_configType (p. 110)	Backup manager configuration information.
}			

Returns:

OK/Error

Description:

To retrieve current backup configuration, use the `get_config` call (p. 139).

Example:

Input

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <set_config>
        <backupm_config>
           <backup_server>
             <protocol>TCP</protocol>
             <address>192.168.0.40</address>
             <login>
              <name>agent</name>
             </login>
             <password>1q2w3e</password>
             <port>4433</port>
           </backup_server>
         </backupm_config>
      </set_config>
    </backupm>
  </data>
</packet>
```

Output

```
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="8ec4ae0556et26car2c8" time="2009-10-22T12:50:20+0000">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient1462-a6f1d061-8bcd-8ec7-1a73-b078fe2d416f</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:backupm>
        <ns1:ok/>
    </ns2:backupm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

get_config

Summary:

Retrieves the default backup configuration.

Request specification:

Name	Min/Max	Type	Description
`get_config`	`1..1`	`none`	
`{`			
`local`	`0..1`	`none`	If this element is present, the call returns the local configuration. If no local configuration is available, returns an error (`ERROR_VZL_NO_CONFIG=308`). If the element is omitted, the call returns the actual configuration (retrieved from the master node if necessary).
`}`			

Returns:

Name	Min/Max	Type	Description
`backupm_config`	`1..1`	`backupm_configType` (p. 110)	Backup manager configuration.

Description:

The default backup configuration is stored in the Agent configuration file. It defines such parameters as backup server location and connectivity, backup type, compression level, certain assumptions and restrictions. The `get_config` call retrieves the currently defined backup configuration. The configuration can be modified using the `set_config` call (p. 137).

Example:

Input

```
<packet version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <get_config/>
    </backupm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/backupm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="84c4ae04f51t798br2c8" time="2009-10-22T12:26:17+0000">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient65-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:backupm>
        <ns2:backupm_config>
            <ns2:backup_server xsi:type="ns3:connection_infoType">
                <ns3:protocol>agent</ns3:protocol>
                <ns3:login xsi:type="ns3:auth_nameType">
                    <ns3:realm>00000000-0000-0000-0000-
000000000000</ns3:realm>
                </ns3:login>
                <ns3:address>local</ns3:address>
            </ns2:backup_server>
            <ns2:chain_length>0</ns2:chain_length>
            <ns2:chain_days>0</ns2:chain_days>
            <ns2:keep_max>0</ns2:keep_max>
            <ns2:compression>1</ns2:compression>
            <ns2:type>0</ns2:type>
            <ns2:pe_backups_limit>1</ns2:pe_backups_limit>
        </ns2:backupm_config>
    </ns2:backupm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

computerm

Purpose:

Computer management interface. Provides a set of calls that allow to manage physical servers and Virtuozzo Containers as if they were regular physical machines. The interface provides limited control over the Hardware Node and is normally used to retrieve the computer system information.

Types

diskType

Summary:

Contains information about hard disk partitions.

Type specification:

Name	Min/Max	Type	Description
`partition`	`0..[]`	`partitionType (p. 141)`	Partition information.

partitionType

Summary:

Contains hard disk partition information.

Type specification:

Name	Min/Max	Type	Description
`name`	`0..1`	`string`	Partition name.
`mount_point`	`0..1`	`string`	Partition mount point.
`fs_type`	`0..1`	`string`	Filesystem type.
`block_size`	`0..1`	`long`	Block size.
`blocks`	`0..1`	`usageType (p. 60)`	Partition disk space information (total/used/free).
`inodes`	`0..1`	`usageType (p. 60)`	Partition disk inodes information (total, used, free).
`option`	`0..[]`	`string`	Mount options.

systemType

Summary:

Contains computer system information.

Type specification:

Name	Min/Max	Type	Description
architecture	1..1	string	Architecture.
os	1..1	osType (p. 50)	Operating system information.
cpu	1..1	cpuType (p. 31)	CPU information.
memory	1..1	resourceType (p. 55)	Physical memory information.
swap	1..1	resourceType (p. 55)	Swap memory information.

Calls

Calls

Call	Description
get_disk (p. 142)	Retrieves disk and partition information.
get_system (p. 145)	Retrieves system information.
get_network (p. 146)	Retrieves network information.
reboot (p. 152)	Reboots or shuts down a server.
get_date (p. 153)	Retrieves current date and time from a server.
set_date (p. 154)	Sets date and time for a server.
get_zones_info (p. 155)	Gets information about known time zones.

get_disk

Summary:

Retrieves information about disks and partitions.

Request specification:

Name	Min/Max	Type	Description
get_disk	0..1	None	

Returns:

Name	Min/Max	Type	Description
disk	0..[]	diskType (p. 141)	Disk information.

Description:

The call retrieves disk and partition information from the current server.

NOTE: For the Linux reiser file system, the command always returns negative values for inodes.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>computerm</target>
  <data>
    <computerm>
      <get_disk/>
    </computerm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/computerm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="8c4ae2f73ct18ber500" time="2009-10-24T12:46:52+0000">
<ns1:origin>computerm</ns1:origin>
<ns1:target>vzclient5-0649a879-f36e-fc4f-b821-5358d55cdefd</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:computerm>
        <ns2:disk>
            <ns2:partition>
                <ns2:name>/dev/vzfs</ns2:name>
                <ns2:mount_point>/</ns2:mount_point>
                <ns2:block_size>4096</ns2:block_size>
                <ns2:fs_type>unknown fs [1448756819]</ns2:fs_type>
                <ns2:option>rw</ns2:option>
                <ns2:blocks xsi:type="ns3:usageType">
                    <ns3:total>262144</ns3:total>
                    <ns3:used>169148</ns3:used>
                    <ns3:free>92996</ns3:free>
                </ns2:blocks>
                <ns2:inodes xsi:type="ns3:usageType">
                    <ns3:total>200000</ns3:total>
                    <ns3:used>17236</ns3:used>
                    <ns3:free>182764</ns3:free>
                </ns2:inodes>
            </ns2:partition>
        </ns2:disk>
    </ns2:computerm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

get_system

Summary:

Retrieves system information.

Request specification:

Name	Min/Max	Type	Description
get_system	0..1	None	

Returns:

Name	Min/Max	Type	Description
system	0..1	systemType (p. 142)	System information.

Description:

The call retrieves system information from the current Hardware Node. The result set is organized into a list of structures, each containing the detailed information about a particular system area, such as operating system, CPU, memory, swap file information.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>computerm</target>
  <data>
    <computerm>
      <get_system/>
    </computerm>
  </data>
</packet>
```

Output

```
<packet id="2cc44855eeft4509" version="4.0.0">
  <origin>computerm</origin>
  <data>
    <computerm>
      <system>
        <architecture>x86 Family 15 Model 3 Stepping 8</architecture>
        Linux
          <platform>Windows</platform>
          <version>5.2.3790 Service Pack 1 Build 3790</version>
          <name>Microsoft Windows Server 2003 family</name>
        </os>
        <cpu>
          <units>0</units>
          <bogomips>0</bogomips>
          <mhz>2793</mhz>
          <family>x86 Family 15 Model 3 Stepping 8</family>
          <number>1</number>
          <cores>1</cores>
          <hyperthreads>1</hyperthreads>
          <name>Intel(R) Celeron(R) CPU 2.80GHz</name>
          <model>GenuineIntel</model>
        </cpu>
        <memory>
          <total>402079744</total>
          <used>371613696</used>
          <free>30466048</free>
        </memory>
        <swap>
          <total>865746944</total>
          <used>344260608</used>
          <free>521486336</free>
        </swap>
      </system>
    </computerm>
  </data>
</packet>
```

get_network

Summary:

Retrieves network information.

Request specification:

Name	Min/Max	Type	Description
get_network	0..1	None	

Returns:

Name	Min/Max	Type	Description
network	0..1		
{			
nameserver	0..[]	string	Name servers.
hostname	1..1	string	Hostname.
default_gateway	1..1	string	Default gateway.
interface	0..[]	interfaceType (p. 41)	Network interface info.
}			

Description:

The call retrieves network settings from the Hardware Node.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>computerm</target>
  <data>
    <computerm>
      <get_network/>
    </computerm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/computerm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="13c4ae2fbaet41bbr9c0" time="2009-10-24T13:06:21+0000">
<ns1:origin>computerm</ns1:origin>
<ns1:target>vzclient11-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:computerm>
        <ns2:network>
            <ns2:hostname>mccp41</ns2:hostname>
            <ns2:default_gateway>0.0.0.0</ns2:default_gateway>
            <ns2:nameserver>10.31.0.1</ns2:nameserver>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>Software Loopback Interface 1</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (SSTP)</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (L2TP)</ns3:name>
```

```
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (PPTP)</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (PPPOE)</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (IPv6)</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (Network Monitor)</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
```

```
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (IP)</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>RAS Async Adapter</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>Broadcom NetXtreme Gigabit Ethernet #2</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>2590058</ns3:packets>
                        <ns3:bytes>2164088371</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>1495949</ns3:packets>
                        <ns3:bytes>405240478</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:ipaddress>10.31.252.35</ns3:ipaddress>
                <ns3:flags>0</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>isatap.{03C77D5A-BEC9-4520-879E-
F9AC9D242A6F}</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>isatap.qa.sw.ru</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
```

```
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (IPv6)-QoS Packet Scheduler-
0000</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (IP)-QoS Packet Scheduler-
0000</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>WAN Miniport (Network Monitor)-QoS Packet Scheduler-
0000</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
                <ns3:name>Broadcom NetXtreme Gigabit Ethernet-QoS Packet
Scheduler-0000</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>0</ns3:packets>
                        <ns3:bytes>0</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
            <ns2:interface xsi:type="ns3:interfaceType">
```

```
                <ns3:name>Broadcom NetXtreme Gigabit Ethernet #2-QoS Packet
Scheduler-0000</ns3:name>
                <ns3:transfer>
                    <ns3:input>
                        <ns3:packets>2590058</ns3:packets>
                        <ns3:bytes>2164088371</ns3:bytes>
                    </ns3:input>
                    <ns3:output>
                        <ns3:packets>1495949</ns3:packets>
                        <ns3:bytes>405240478</ns3:bytes>
                    </ns3:output>
                </ns3:transfer>
                <ns3:flags>1</ns3:flags>
            </ns2:interface>
        </ns2:network>
    </ns2:computerm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

reboot

Summary:

Reboots or shuts down the machine.

Request specification:

Name	Min/Max	Type	Description
reboot	0..1	none	
{			
shutdown	0..1	none	If present, the server will be shut down.
}			

Returns:

OK/ERROR.

Description:

The call reboots or shuts down the current Hardware Node. To perform a complete shutdown, include the optional `shutdown` element. To reboot, include just the `reboot` element. All users that are currently logged on to the server will be automatically logged off.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>computerm</target>
  <data>
    <computerm>
      <reboot>
        <shutdown/>
      </reboot>
    </computerm>
  </data>
</packet>
```

get_date

Summary:

Retrieves date and time information from the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
get_date	0..1	None	

Returns:

Name	Min/Max	Type	Description
date	1..1		
{			
time	0..1	datetime_type	Current time.
time_zone	0..1	string	Time zone information.
}			

Example:

Input

```
<packet>
  <target>computerm</target>
  <data>
    <computerm>
      <get_date/>
    </computerm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/computerm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="15c4ae30026t1ebr9c0" time="2009-10-24T13:25:25+0000">
<ns1:origin>computerm</ns1:origin>
<ns1:target>vzclient11-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:computerm>
        <ns2:date>
            <ns2:time>2009-10-24T13:25:25+0000</ns2:time>
            <ns2:time_zone>Russian Standard Time</ns2:time_zone>
        </ns2:date>
    </ns2:computerm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

set_date

Summary:

Sets local date and time on the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
set_date			
{			
date	1..1		
{			
year	1..1	int	Four-digit year.
month	1..1	int	Month (1 to 12).
day	1..1	int	Day of month (1 to 31).
hour	1..1	int	Hour (0 to 23).
minute	1..1	int	Minute (0 to 59).

second	1..1	int	Second (0 to 59).
day_of_week	0..1	int	Day of week (0 to 7; 0 and 7 both refer to Sunday).
}			
{			
time_zone	0..1	string	Time zone information.
}			
}			

Returns:

OK/Error

Example:

```
<packet>
  <target>computerm</target>
  <data>
    <computerm>
      <set_date>
        <date>
          <year>2007</year>
          <month>1</month>
          <day>23</day>
          <hour>3</hour>
          <minute>2</minute>
          <second>25</second>
          <day_of_week>2</day_of_week>
          <time_zone>America/New_York</time_zone>
        </date>
      </set_date>
    </computerm>
  </data>
</packet>
```

get_zones_info

Summary:

Retrieves information about known time zones.

Request specification:

Name	Min/Max	Type	Description
get_zones_info	0..1	None	

Returns:

Name	Min/Max	Type	Description
time_zone	0..[]		
{			
name	1..1	string	Zone name.
display_name	1..1	string	Display name.
}			

Example:

Input

```
<packet>
  <target>computerm</target>
  <data>
    <computerm>
      <get_zones_info/>
    </computerm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/computerm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="14c4ae2fe3bt26e9r9c0" time="2009-10-24T13:17:15+0000">
<ns1:origin>computerm</ns1:origin>
<ns1:target>vzclient11-2fabe194-0477-0871-a32d-49220050cf5c</ns1:target>
<ns1:dst>
    <director>gend</director>
</ns1:dst>
<ns1:data>
    <ns2:computerm>
        <ns2:time_zone>
            <ns2:name>Afghanistan Standard Time</ns2:name>
            <ns2:display_name>(GMT+04:30) Kabul</ns2:display_name>
        </ns2:time_zone>
        <ns2:time_zone>
            <ns2:name>Alaskan Standard Time</ns2:name>
            <ns2:display_name>(GMT-09:00) Alaska</ns2:display_name>
        </ns2:time_zone>
        <ns2:time_zone>
            <ns2:name>Arab Standard Time</ns2:name>
            <ns2:display_name>(GMT+03:00) Kuwait, Riyadh</ns2:display_name>
        </ns2:time_zone>
    <!-- The complete output is omitted for brevity -->
        <ns2:time_zone>
            <ns2:name>Yakutsk Standard Time</ns2:name>
            <ns2:display_name>(GMT+09:00) Yakutsk</ns2:display_name>
        </ns2:time_zone>
    </ns2:computerm>
</ns1:data>
<src>
    <director>gend</director>
</src>
</packet>
```

data_storagem

Purpose:

The base data storage management interface.

Derived interfaces:

`backup_storagem` (p. 106)

Types

ds_locationType

Summary:

Contains data storage location configuration

Type specification:

Name	Min/Max	Type	Description
`type`	`1..1`	`int`	Data storage location type: `0` -- local `1` -- Windows share.
`path`	`1..1`	`base64Binary`	Root data storage path.
`login`	`0..1`	`auth_nameType` (p. 29)	Login for remote storage.
`password`	`0..1`	`string`	Password for remote storage.

ds_configurationType

Summary:

Describes data storage configuration.

Type specification:

Name	Min/Max	Type	Description
`location`	`1..1`	`ds_locationType` (p. 157)	Data storage location.

ds_object_infoType

Summary:

Displays information about an object in data storage.

Type specification:

Name	Min/Max	Type	Description
`time`	`1..1`	`datetime_type` (p. 25)	Object modification or creation time.
`size`	`1..1`	`long`	Object size in storage.
`type`	`1..1`	`int`	Object type. Values depend on implementation.
`id`	`1..1`	`ds_object_path_type` (p. 158)	Object ID/path.
`storage_eid`	`1..1`	`eid_type` (p. 25)	Server ID of the data storage server.
`info`	`1..1`	`infoType` (p. 38)	Additional information about an object.

ds_object_path_type

Summary:

Object identifier.

Type specification:

Restriction: `string`.

Calls

Call	Description
get_storage_config (p. 159)	Retrieves data storage configuration.
set_storage_config (p. 159)	Sets data storage configuration.

get_storage_config

Summary:

Retrieves data storage configuration.

Request specification:

Name	Min/Max	Type	Description
get_storage_config	1..1	none	

Returns:

Name	Min/Max	Type	Description
config		ds_configurationType (p. 157)	Data Storage configuration.

set_storage_config

Summary:

Sets the data storage configuration.

Request specification:

Name	Min/Max	Type	Description
set_storage_config	1..1		
{			
config	1..1	ds_configurationType (p. 157)	New Data Storage configuration information.
}			

Returns:

OK/Error

devm

Purpose:

The base device management interface. Supported virtualization technologies have their own device management interfaces, which are derived from the `devm` interface.

Note: At the time of this writing, the only supported virtualization technology is Virtuozzo Containers.

Derived interfaces:

`vzadevm` (p. 579)

Types

mount_deviceType

Summary:

Contains device mount information.

Type specification:

Name	Min/Max	Type	Description
`permanent`	`0..1`	`boolean`	Mount type: `true` -- permanent mount. `false` -- temporary mount. Permanently mounted devices are mounted automatically at system boot. Please note that when mounting a device on the Hardware Node, the mount information is written into the `/etc/fstab` file. However, when mounting a device inside a Virtuozzo Container, the mount information is written into one of the Virtuozzo script files, which are executed at the time a specific Container is started.
`device`	`0..1`	`string`	On Linux, the name of the device. On Windows, the name of the device or an EFD image file.

point	1..1	string	Mount point. On Linux , it is the directory name. On Windows, it is the drive letter, e.g. E:
filesystem	0..1	string	Filesystem type.
active	0..1	boolean	Mount status: true -- the mount is active. false -- the mount is inactive. The "inactive" status applies only to the permanent mounts. Temporary mounts can exist only in the "active" state. In other words, you cannot mount a device temporarily and make it inactive at the same time. If you try to "deactivate" a temporarily mounted device, the mount will be deleted altogether.
size	0..1	long	The size of the EFD image file. When creating an EFD image, specify the desired size here.
interface	0..1	string	Interface type: SCSI -- SCSI storage device. This element is used when configuring Windows Cluster Server (MSCS) support in a Virtuozzo Container. You can create an EFD image, mount it inside a Virtuozzo Container, and configure the new drive to be recognized by Windows running in the Container as a SCSI storage device. In order to do that, include the interface element containing the SCSI value. If this element is omitted while mounting an EFD image inside a Container, the image is mounted as a common loopback.

common_deviceType

Summary:

Common device information.

Type specification:

Name	Min/Max	Type	Description
name	0..1	string	Device name.
description	0..1	string	Device description.

Subtypes:

`windows_deviceType (p. 162)`

`scsi_deviceType` (p. 162)

windows_deviceType

Summary:

Contains information about any non-SCSI device on Windows.

Type specification:

Extends `common_deviceType (p. 161)`

Adds the following elements:

Name	Min/Max	Type	Description
`physical_name`	`0..1`	`base64Binary`	Physical device name.

scsi_deviceType

Summary:

Contains information about a SCSI device.

Type specification:

Extends `windows_deviceType (p. 162)`

The type has no additional elements.

Calls

Call	Description
get_mounts (p. 172)	Retrieves a list of devices mounted inside the specified server.
new_mount (p. 180)	Mounts the specified device.
umount (p. 186)	Unmounts the specified device.
get_info (p. 171)	Gets information about all available filesystems, partitions and devices.
create_drive (p. 163)	Creates a new file system image file and loopback-mounts it in the specified server.
delete_drive (p. 167)	Unmounts and deletes the file system image file.
resize_drive (p. 185)	Resizes file system image file.
format_drive (p. 168)	Formats a disk drive.
list_devices (p. 176)	Lists devices available on the Hardware Node.

forward_device (p. 169)	Makes a SCSI device on the Hardware Node visible and accessible to Virtuozzo Containers.
remove_forward	Cancels forwarding of a device (see `forward_device` above).
list_forward (p. 178)	List the device forwarding information (see `forward_device` above).

create_drive

This call is available on Windows only.

Summary:

Creates a new file system image and loopback-mounts it on the specified server.

Request specification:

Name	Min/Max	Type	Description
`create_drive`		`mount_deviceType (p. 160)`	
`{`			
`eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`}`			

Returns:

Name	Min/Max	Type	Description
`device_info`			New disk information.
`{`			
`device`	`0..[]`	`string`	The full path to the image file.
`}`			

Description:

Use the `create_drive` call to create EFD file system images. EFD is SWsoft's proprietary file system. The call creates a new, empty file system image file and then loopback-mounts it the specified Container. After the file is created and mounted, you can use it just like any other physical disk drive inside the Container.

The `device` parameter specifies the name and path where the file will reside. If you specify a full path, it will be treated as a path on the Hardware Node. If you specify just the name of the file, the file will be created in the Container private area. If you omit the `device` element, the file name will be generated automatically and the file will be created in the Container private area.

The image located in a particular Container private area can be mounted inside that Container only. Note that when creating an image file, you have to mount it inside a Container, you cannot create an unmounted EFD image.

The following describes the parameters used in this call:

`device` -- The name and path of the image file. If not specified, the name will be generated automatically by using the `lpbk` prefix (meaning "loopback") followed by a numeric value indicating the file number (i.e. `0000`, `0001`, `0002`, etc.), and the `.efd` extension. The following are the examples of the automatic file names: `lpbk0000.efd`, `lpbk0001.efd`, `lpbk0002.efd`. If the name doesn't contain a full path, the image file will be created in the Container private area. If the full path is specified (e.g. `C:\img.efd`), it will be treated as a path on the Hardware Node.

`size` -- The size of the image file in bytes.The minimum allowed size is 6 megabytes. There's no limit on the max size. You can change this size later with the `resize_drive` call (p. 185).

`point` -- The mount point. This must be the disk drive letter, e.g. `E:`

`eid` -- The ID of the server where you would like to mount the image file.

Configuring shared loopback based Container clustering

`interface` -- To configure shared loopback based Container clustering, the new drive must be recognized by Windows running inside the Container as SCSI block device. This is accomplished by include the `interface` element containing the `SCSI` value. Before you can start using the drive inside the Container, you'll have to initialize it as follows:

1. Log in to the Container via Remote Desktop.
2. On the Windows Start menu, select Programs->Administrative Tools->Computer Management.
3. In the Computer Management window, select Storage->Disk Management.
4. Right-click on the new disk and select the Initalize option from the pop-up menu.
5. Right-click on the disk partition and select the New Parition option from the menu.

6 Complete the New Partition wizard using the default values on all screens except the Assign the Drive Letter or Path screen where you should select the drive letter that you specified in the `point` parameter of the `create_drive` call.

The following steps describe how to deploy shared loopback based Container clustering.

1. Enable Windows Cluster Server support in a Virtuozzo Container by turning the `failover_cluster` capability on. The following code example shows how it is accomplished:

```
<packet version="4.0.0">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set>
        <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
        <config>
          <capability>
            <id>failover_cluster</id>
            <value>1</value>
          </capability>
        </config>
      </set>
    </vzaenvm>
  </data>
</packet>
```

2. Create a shared disk drive as described above.

3. Login to the Container via Remote Desktop and create/add the Container to a cluster using Microsoft Cluster Administrator.

Example 1:

Creating the filesystem image named `img.efd` in the `C:` drive root directory on the Hardware Node and mounting it as an `F:` drive inside the specified Virtuozzo Container.

Input

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <create_drive>
        <device>C:\img.efd</device>
        <point>F:</point>
        <size>6000000</size>
        <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
      </create_drive>
    </vzadevm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzadevm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="2bc46486e35t6443re80" time="2007-05-15T13:12:23+0000" priority="0"
version="4.0.0">
```

```
<ns1:origin>vzadevm</ns1:origin>
<ns1:target>vzclient2</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzadevm>
    <ns2:device_info>
      <ns2:device>C:\img.efd</ns2:device>
    </ns2:device_info>
  </ns2:vzadevm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 2:

In this example we do not specify the name for the image file. The name, therefore, will be generated automatically and the file will be created in the Container private area.

Input

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <create_drive>
        <point>I:</point>
        <size>6000000</size>
        <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
      </create_drive>
    </vzadevm>
  </data>
</packet>
```

Output

The output below contains the name of the image file: `lpbk0004.efd`

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzadevm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="2cc46487129t66bbre80" time="2007-05-15T13:24:59+0000" priority="0"
version="4.0.0">
<ns1:origin>vzadevm</ns1:origin>
<ns1:target>vzclient2</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzadevm>
    <ns2:device_info>
      <ns2:device>lpbk0004.efd</ns2:device>
    </ns2:device_info>
  </ns2:vzadevm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 3:

This example demonstrates how to create a shared SCSI storage device inside a Container.

Input

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <create_drive>
        <device>C:\img007.efd</device>
        <point>L:</point>
        <size>6000000</size>
        <interface>SCSI</interface>
        <eid>715d6510-b7f1-4eda-98e2-3c6b6ee1f608</eid>
      </create_drive>
    </vzadevm>
  </data>
</packet>
```

delete_drive

This call is available on Windows only.

Summary:

Unmounts and deletes the specified filesystem image file.

Request specification:

Name	Min/Max	Type	Description
`delete_drive`			
`{`			
`eid`	`0..1`	`eid_type` (p. 25)	Server ID.
`[`			Denotes a choice between the `point` and the `device` elements.
`point`	`1..1`	`string`	Mount point.
`device`	`1..1`	`string`	*This parameter is not currently used.*
`]`			
`}`			

Returns:

OK/Error

Description:

The `delete_drive` call unmounts the EFD image from the specified Container that was previously mounted using the `create_drive` (p. 163) or the `new_mount` (p. 180) calls. The image file is then permanently deleted from the Hardware Node or the Container private area. CAUTION! You will not be able to re-mount the file and recover its contents after executing this call. To unmount the file system image without deleting the image file, use the `umount` call (p. 186).

Example:

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <delete_drive>
        <eid>715d6510-b7f1-4eda-98e2-3c6b6ee1f608</eid>
        <point>K:</point>
      </delete_drive>
    </vzadevm>
  </data>
</packet>
```

format_drive

This call has not been implemented in the current Agent version.

Summary:

Formats a disk drive.

Request specification:

Name	Min/Max	Type	Description
`format_drive`			
`{`			
`device`	`1..1`	`string`	Device name.
`type`	`1..1`	`string`	Filesystem type to create.
`label`	`0..1`	`string`	Volume label.
`block_size`	`0..1`	`long`	Filesystem block size (in bytes).
`}`			

forward_device

This call is currently available on Windows only.

Summary:

Makes a SCSI device on the Hardware Node accessible from within a Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
`forward_device`			
`{`			
`forward`	`1..1`		
`{`			
`source`	`1..1`		Source device information.
`{`			
`eid`	`0..1`	`eid_type (p. 25)`	*Not used here.*
`device`	`1..1`	`common_deviceType (p. 161)`	The device information. Use `scsi_deviceType` type (p. 162) -- the the subtype of `common_deviceType (p. 161)` -- when populating this structure.
`}`			
`destination`	`1..1`		Target device information.
`{`			
`eid`	`0..1`	`eid_type (p. 25)`	Target server ID.
`device`	`1..1`	`common_deviceType (p. 161)`	The device info as you want to be displayed in the target server.
`}`			
`}`			
`}`			

Returns:

OK/Error

Description:

This functionality exists for the purpose of setting up *SAN (Storage Area Networks) based Container clustering*. One of the requirements for setting up SAN based clustering is the ability to access remote storage devices (shared SCSI, fiberchannel, etc.) from within a Container by mounting such a device inside the Container. The `forward_device` call allows to accomplish this task. The source device information must include the name and the ID of the device (the ID is contained in the `physical_name` element). To retrieve the list of SCSI devices available on the Hardware Node, use the `list_device` call (p. 176) and select the entries from the result set that are contained in the `device` element of type `scsi_deviceType` (p. 162). The following is an example of such an entry:

```
<ns2:device xsi:type="ns2:scsi_deviceType">
  <ns2:name>S SCSI Disk Device</ns2:name>
  <ns2:description>Disk drive</ns2:description>
  <ns2:physical_name>
       U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0Rf
       Vk1XQVJFX1ZJUlRVQUxfUyZSRVZfMS4wXDQm
       M0E3Mzk1MjkmMCYwMDA=
  </ns2:physical_name>
</ns2:device>
```

The target device information must include the ID of the server where you would like to mount the drive, and the device info (name, description) as you want it to be displayed in the target server.

Example:

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/devm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <forward_device>
        <forward>
          <source>
            <ns2:device xsi:type="ns2:scsi_deviceType">
              <ns2:name>S SCSI Disk Device</ns2:name>
              <ns2:description>Disk drive</ns2:description>

<ns2:physical_name>U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0RfVk1XQVJFX1ZJUlRVQUxfUyZ
SRVZfMS4wXDQmM0E3Mzk1MjkmMCYwMDA=</ns2:physical_name>
            </ns2:device>
          </source>
          <destination>
            <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
            <ns2:device xsi:type="ns2:scsi_deviceType">
               <ns2:name>My SCSI Disk Device</ns2:name>
               <ns2:description>Disk drive</ns2:description>
            </ns2:device>
          </destination>
        </forward>
      </forward_device>
    </vzadevm>
  </data>
```

```
</packet>
```

get_info

Summary:

Retrieves information about all available filesystems, partitions, and devices from the current server.

Request specification:

Name	Min/Max	Type	Description
get_info	0..1	none	

Returns:

Name	Min/Max	Type	Description
device_info			
{			
filesystem	0..[]	string	Filesystem type.
device	0..[]	string	Device name.
partition	0..[]	string	Partition name.
}			

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <get_info/>
    </vzadevm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzadevm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="22c4649b5f1t26e9rf74" time="2007-05-15T16:57:31+0000" priority="0"
version="4.0.0">
<ns1:origin>vzadevm</ns1:origin>
<ns1:target>vzclient5-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzadevm>
    <ns2:device_info>
      <ns2:partition>/dev/sda1</ns2:partition>
      <ns2:partition>/dev/sda2</ns2:partition>
      <ns2:partition>/dev/sda3</ns2:partition>
      <ns2:partition>/dev/sda5</ns2:partition>
      <ns2:partition>/dev/dm-0</ns2:partition>
      <ns2:filesystem>auto</ns2:filesystem>
    </ns2:device_info>
  </ns2:vzadevm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_mounts

Summary:

Retrieves information about the filesystems mounted on the specified server.

Request specification:

Name	Min/Max	Type	Description
`get_mounts`			
`{`			
`eid`	`0..1`	`eid_type` (p. 25)	Server ID. If omitted, a list of file systems mounted in the current server will be retrieved.
`}`			

Returns:

Name	Min/Max	Type	Description
mount			Mount information.
{			
permanent	0..1	boolean	Contains true if the filesystem is mounted permanently. Contains false otherwise. Permanently mounted filesystems have corresponding entries in /etc/fstab file and are mounted automatically every time the system reboots.
device	0..1	string	Device name, e.g. /dev/fd0
point	1..1	string	Mount point, e.g. /mnt/floppy
filesystem	0..1	string	Filesystem type.
active	0..1	boolean	Contains true if mount is active. Contains false otherwise. Temporarily mounted filesystems exist only in the "active" state. Trying to make the temporary mount inactive will remove the mount from the system.
}			

Returns:

Name	Min/Max	Type	Description
mount			Mount information.
{			
device	0..1	string	Device name. Can be a letter of the physical drive on the Hardware Node or the name of the loopback-mounted filesystem image.
point	1..1	string	The drive letter the filesystem is mounted to.
size	0..1	long	The total size of the media in the mounted physical drive or the size of the loopback-mounted file, in bytes.
}			

Description:

On Linux, the `get_mounts` call retrieves information about currently mounted filesystems (usually a device, but can also be a directory name or a dummy).

On Windows, it is possible to mount a physical drive or a filesystem image. Depending on the source, the `get_mounts` call returns the following information:

- If the device is a physical drive on the Hardware Node, the `device` parameter will contain the drive letter. For example, if a Hardware Node has a CD-ROM drive with a letter `D:` assignment, and this drive is mounted on a server as drive `F:`, the `device` parameter will contain the `D:` and the `point` parameter will contain the `F:`. The `size` parameter will contain the total size of the media in the drive (hard disk partition, floppy disk, CD-ROM, etc.).
- If the device is a loopback-mounted filesystem image, the `device` parameter will contain the name and path of the file and the `point` parameter will contain the drive letter that the image file is mounted to. The `size` parameter will contain the file size in bytes.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <get_mounts>
        <eid>ba17a0c5-9036-473c-a813-aa6f5b36cf16</eid>
      </get_mounts>
    </vzadevm>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>devm</ns1:origin>
<ns1:target>vzclient5</ns1:target>
<ns1:data>
  <ns2:devm>
    <ns2:mount>
      <ns2:device>/dev/VolGroup00/LogVol00</ns2:device>
      <ns2:filesystem>ext3</ns2:filesystem>
      <ns2:permanent>1</ns2:permanent>
      <ns2:active>1</ns2:active>
      <ns2:point>/</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>LABEL=/boot</ns2:device>
      <ns2:filesystem>ext3</ns2:filesystem>
      <ns2:permanent>1</ns2:permanent>
      <ns2:point>/boot</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>/dev/sdb1</ns2:device>
      <ns2:filesystem>auto</ns2:filesystem>
      <ns2:permanent>1</ns2:permanent>
      <ns2:point>/vz</ns2:point>
    </ns2:mount>
    <ns2:mount>
```

```
      <ns2:device>/dev/hdd</ns2:device>
      <ns2:filesystem>auto</ns2:filesystem>
      <ns2:point>/media/cdrom</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>/dev/hdc</ns2:device>
      <ns2:filesystem>auto</ns2:filesystem>
      <ns2:point>/media/cdrom1</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>/dev/hdb</ns2:device>
      <ns2:filesystem>auto</ns2:filesystem>
      <ns2:point>/media/cdrom2</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>/dev/hda</ns2:device>
      <ns2:filesystem>auto</ns2:filesystem>
      <ns2:point>/media/cdrom3</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>/dev/fd0</ns2:device>
      <ns2:filesystem>auto</ns2:filesystem>
      <ns2:point>/media/floppy</ns2:point>
    </ns2:mount>
  </ns2:devm>
</ns1:data>
</ns1:packet>
```

Example:

Input

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <get_mounts>
        <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
      </get_mounts>
    </vzadevm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzadevm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="2dc464871d0t428bre80" time="2007-05-15T13:27:38+0000" priority="0"
version="4.0.0">
<ns1:origin>vzadevm</ns1:origin>
<ns1:target>vzclient2</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzadevm>
    <ns2:mount>
      <ns2:device>root.efd</ns2:device>
      <ns2:size>174094336</ns2:size>
      <ns2:point>C:</ns2:point>
    </ns2:mount>
    <ns2:mount>
      <ns2:device>C:\img.efd</ns2:device>
      <ns2:size>6027264</ns2:size>
      <ns2:point>F:</ns2:point>
```

```
    </ns2:mount>
    <ns2:mount>
      <ns2:device>lpbk0000.efd</ns2:device>
      <ns2:size>1000011776</ns2:size>
      <ns2:point>K:</ns2:point>
    </ns2:mount>
  </ns2:vzadevm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

list_device

This call is available on Windows only.

Summary:

Lists devices installed on the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
`list_device`			
`{`			
`device`	`0..[]`	`common_deviceType (p. 161)`	Device info. If none specified, returns a list of all available devices.
`}`			

Returns:

Name	Min/Max	Type	Description
`device`	`0..1`	`common_deviceType (p. 161)`	Device information. The actual data type returned here can be used to identify the device type (SCSI, other Windows devices). See the subtypes of the `common_deviceType (p. 161)` for the available types. When setting up Container clustering, select the desired SCSI device from the list.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>devm</target>
  <data>
    <devm>
      <list_device/>
    </devm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/devm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="55c464dc9d9t2350r878" time="2007-05-19T14:44:18+0000" priority="0"
version="4.0.0">
<ns1:origin>devm</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:devm>
    <ns2:device xsi:type="ns2:windows_deviceType">
      <ns2:name>Microsoft AC Adapter</ns2:name>
      <ns2:description>Microsoft AC Adapter</ns2:description>
      <ns2:physical_name>QUNQSVxBQ1BJMDAwM1wx</ns2:physical_name>
    </ns2:device>
    <ns2:device xsi:type="ns2:windows_deviceType">
      <ns2:name>ACPI Fixed Feature Button</ns2:name>
      <ns2:description>ACPI Fixed Feature Button</ns2:description>

<ns2:physical_name>QUNQSVxGSVhFREJVVFRPTlwyJkRBQkEzRkYmMA==</ns2:physical_name
>
    </ns2:device>
    <ns2:device xsi:type="ns2:windows_deviceType">
      <ns2:name>Microsoft System Management BIOS Driver</ns2:name>
      <ns2:description>Microsoft System Management BIOS
Driver</ns2:description>
      <ns2:physical_name>Uk9PVFxTWVNURU1cMDAwMg==</ns2:physical_name>
    </ns2:device>
    <ns2:device xsi:type="ns2:scsi_deviceType">
      <ns2:name>VMware, VMware Virtual S SCSI Disk Device</ns2:name>
      <ns2:description>Disk drive</ns2:description>

<ns2:physical_name>U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0RfVk1XQVJFX1ZJUlRVQUxfUyZ
SRVZfMS4wXDQmM0E3Mzk1MjkmMCYwMDA=</ns2:physical_name>
    </ns2:device>
    <ns2:device xsi:type="ns2:scsi_deviceType">
      <ns2:name>VMware, VMware Virtual S SCSI Disk Device</ns2:name>
      <ns2:description>Disk drive</ns2:description>

<ns2:physical_name>U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0RfVk1XQVJFX1ZJUlRVQUxfUyZ
SRVZfMS4wXDQmM0E3Mzk1MjkmMCYwMTA=</ns2:physical_name>
    </ns2:device>
    <ns2:device xsi:type="ns2:windows_deviceType">
      <ns2:name>Generic volume</ns2:name>
      <ns2:description>Generic volume</ns2:description>

<ns2:physical_name>U1RPUkFHRVxWT0xVTUVcMSYzMEE5NjU5OCYwJlNJR05BVFVSRUM2NzhERTk
xT0ZGU0VUN0UwMExFTkdUSDNGRkFCRDIwMA==</ns2:physical_name>
    </ns2:device>
```

```
    <ns2:device xsi:type="ns2:windows_deviceType">
      <ns2:name>Generic volume</ns2:name>
      <ns2:description>Generic volume</ns2:description>

<ns2:physical_name>U1RPUkFHRVxWT0xVTUVcMSYzMEE5NjU5OCYwJlNJR05BVFVSRUVGMThFRjE
4T0ZGU0VUN0UwMExFTkdUSDFGRjU4MjgwMA==</ns2:physical_name>
    </ns2:device>
  </ns2:devm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

list_forward

This call is available on Windows only.

Summary:

Lists the device forwarding information. See the `forward_device` call (p. 169) for more info on device forwarding.

Request specification:

Name	Min/Max	Type	Description
`list_forward`			
`{`			
`  eid`	`0..1`	`eid_type` (p. 25)	The ID of the server for which you would like to see the device forwarding information.
`}`			

Returns:

Name	Min/Max	Type	Description
`forward`	`1..[]`		A list of "forwarded" devices.
`{`			
`  source`	`1..1`		Source device information.
`  {`			
`    eid`	`0..1`	`eid_type` (p. 25)	Host server ID.
`    device`	`0..1`	`common_deviceType` (p. 161)	Device information.
`  }`			
`  destination`			Destination device information.
`  {`			
`    eid`	`0..1`	`eid_type` (p. 25)	Server ID.

device	0..1	common_deviceType (p. 161)	Device information.
}			
}			

Example:

Input

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <list_forward>
        <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
      </list_forward>
    </vzadevm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/devm"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzadevm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="51c464db67ft56aer878" time="2007-05-19T13:21:44+0000" priority="0"
version="4.0.0">
<ns1:origin>vzadevm</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzadevm>
    <ns2:forward>
      <ns2:source>
        <ns2:device xsi:type="ns3:windows_deviceType">
          <ns3:name>S SCSI Disk Device</ns3:name>
          <ns3:description>SCSI Disk Device</ns3:description>

<ns3:physical_name>U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0RfVk1XQVJFX1ZJUlRVQUxfUyZ
SRVZfMS4wXDQmM0E3Mzk1MjkmMCYwMDA=</ns3:physical_name>
        </ns2:device>
      </ns2:source>
      <ns2:destination>
        <ns2:eid>7f29d970-3e31-46f3-9b59-2654329e3e55</ns2:eid>
        <ns2:device xsi:type="ns3:windows_deviceType">
          <ns3:name>My SCSI Disk Device</ns3:name>
          <ns3:description>S SCSI Disk Device</ns3:description>

<ns3:physical_name>U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0RfVk1XQVJFX1ZJUlRVQUxfUyZ
SRVZfMS4wXDQmM0E3Mzk1MjkmMCYwMDA=</ns3:physical_name>
        </ns2:device>
      </ns2:destination>
    </ns2:forward>
  </ns2:vzadevm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

new_mount

Summary:

Mounts a Hardware Node device on the Hardware Node or inside a Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
new_mount		mount_deviceType (p. 160)	
{			
eid	1..1	eid_type (p. 25)	Server ID.
}			

Returns:

OK/Error

Description:

The `device` parameter specifies the name of the device that you would like to mount. The `point` parameter is used to specify the name of the directory where you would like to mount the device. If the specified directory does not exist, it will be created. Once the device is mounted, the mount point will refer to the root of the file system on the device.

If the `filesystem` element is absent or set to `auto`, the system will attempt to detect the file system type automatically. It is not always possible to recognize some file systems due to differences in implementations. That's why you might want to specify the file system type via the `filesystem` option.

If the `permanent` parameter is set to `true`, the device will be mounted permanently, which means that the device will be mounted automatically every time the server is started or restarted. If the parameter is set to `false` or omitted, the device will be mounted temporarily, i.e. the mount point will exist for the duration of the current session only. You can change the status of the mount point to "permanent" later if you wish. In order to do that, execute the `new_mount` call again passing the same parameters and values as when you created the mount point except the `permanent` parameter which should be set to `true`, and the `active` parameter which should be omitted from the call (see code examples below). Please note that when mounting a device on the Hardware Node, the mount information is written into the `/etc/fstab` file. However, when mounting a Hardware Node device inside a Virtuozzo Container, the mount information is written into one of the internal Virtuozzo script files, which are executed at the time a specific Container is started. Do not attempt to modify the `fstab` file inside the Container manually as it will not work.

If the `active` element is included, the command will make an attempt to attach the file system on the device to the mount point at the end of the operation. If this option is omitted or set to `false`, the mount point will still be created but it will remain in the "inactive" state, in which case you will have to activate it before it can be used. To activate a mount point (to attach the filesystem on the device to it), use the `new_mount` call again passing the same parameters and values as when you created the mount point except the `active` parameter which should be set to `true`, and the `permanent` parameter which should be omitted from the call (see code examples below). If the device doesn't contain a valid file system (e.g. the CD drive is empty), the mount point will be created but will also remain in the "inactive" state. Note that temporary mounts cannot exist in the "inactive" state. If you specify the `permanent` parameter and the `active` parameter both set to false (or if you omit both parameters), the mount point will not be created.

To retrieve the list of the available filesystems, partitions, and devices, use the `get_info` call (p. 171).

On Windows, the `new_mount` call allows to do the following:

- Mount a physical drive installed on the Hardware Node inside a Virtuozzo Container. The `device` parameter is used to specify the physical drive letter. The `point` parameter is used to specify the drive letter representing the mount point. For example, if a CD-ROM drive is assigned the letter `D:` on the Hardware Node and you would like to mount it to drive letter `F:` on a Virtuozzo Container, the `device` parameter should contain the `D:` value and the `point` parameter should contain the `F:` value.
- Loopback-mount EFD filesystem images. EFD is SWsoft's proprietary filesystem. Use the `device` parameter to specify the image file name and path. The `point` parameter must contain the drive letter that you want to use for this mount. The `interface` parameter must be omitted from the request. Please note that you cannot mount the same image in more than one Container. You can create EFD images using the `create_drive` call (p. 163).
- Configure Windows Cluster Server (MSCS) support. If you have an EFD image file on the Hardware Node (or on a remote network share), you can mount it inside a Virtuozzo Container as an emulated shared SCSI storage device. To mount a shared image, use the `device` parameter to specify the image file name and path on the Hardware Node. The `point` parameter must contain the drive letter that you would like to use for the mount. The value of the `interface` parameter must be `SCSI`, which indicates that you are mounting an image as an emulated shared SCSI storage device. Before you can use the new drive inside the Container, you will have to initialize it (see the `create_drive` call (p. 163) for more information on how to initialize the new drive in Windows and for more info Windows Cluster Server support in Virtuozzo Containers).

Example 1:

Temporarily mounting a Hardware Node partition inside a Container.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <new_mount>
        <device>/dev/sda2</device>
        <point>/mydrive</point>
        <eid>107d1f60-841e-8c43-8152-3c368ef3c366</eid>
        <active>true</active>
      </new_mount>
    </vzadevm>
  </data>
</packet>
```

Example 2:

Changing the status of the existing mount point from "temporary" to "permanent".

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <new_mount>
        <permanent>true</permanent>
        <device>/dev/sda2</device>
        <point>/mydrive</point>
        <eid>107d1f60-841e-8c43-8152-3c368ef3c366</eid>
      </new_mount>
    </vzadevm>
  </data>
```

```
</packet>
```

Example 3:

Changing the state of the existing mount point from "inactive" to "active", i.e. attaching the filesystem on the device to the mount point.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <new_mount>
        <device>/dev/sda2</device>
        <point>/mydrive</point>
        <active>true</active>
        <eid>107d1f60-841e-8c43-8152-3c368ef3c366</eid>
      </new_mount>
    </vzadevm>
  </data>
</packet>
```

Example 4:

Mounting the CD-ROM drive `D:` installed on the Hardware Node inside the specified Container. Assigning the drive letter `F:` to the new mount inside the Container.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <new_mount>
        <device>C:\</device>
        <point>F:</point>
        <eid>b85f10fc-e42b-4e1c-a18b-85c6a25501b8</eid>
      </new_mount>
    </vzadevm>
  </data>
</packet>
```

Example 5:

Mounting an EFD filesystem image inside a Container.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <new_mount>
        <device>C:\img005.efd</device>
        <point>M:</point>
        <eid>715d6510-b7f1-4eda-98e2-3c6b6ee1f608</eid>
      </new_mount>
    </vzadevm>
  </data>
</packet>
```

remove_forward

This call is available on Windows only.

Summary:

Cancels the forwarding of a device that was added to a Virtuozzo Container using the `forward_device` call (p. 169).

Request specification:

Name	Min/Max	Type	Description
remove_forward			
{			
forward	1..1		
{			
source	1..1		The original source device information.
{			
eid	0..1	eid_type (p. 25)	*Not used here.*
device	1..1	common_deviceType (p. 161)	Device information.
}			
destination			The device information inside a Container.
{			
eid	0..1	eid_type (p. 25)	Server ID.
device	1..1	common_deviceType (p. 161)	Device information.
}			
}			
}			

Returns:

OK/Error

Description:

The `remove_forward` call has essentially the same exact parameters and values as the `forward_device` call (p. 169). The only difference between the two is the name of the call itself.

Example:

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/devm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <remove_forward>
        <forward>
          <source>
            <ns2:device xsi:type="ns2:scsi_deviceType">
              <ns2:name>S SCSI Disk Device</ns2:name>
              <ns2:description>Disk drive</ns2:description>

<ns2:physical_name>U0NTSVxESVNLJlZFTl9WTVdBUkVfJlBST0RfVk1XQVJFX1ZJUlRVQUxfUyZ
SRVZfMS4wXDQmM0E3Mzk1MjkmMCYwMDA=</ns2:physical_name>
            </ns2:device>
          </source>
          <destination>
            <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
            <ns2:device xsi:type="ns2:scsi_deviceType">
               <ns2:name>My SCSI Disk Device</ns2:name>
               <ns2:description>Disk drive</ns2:description>
            </ns2:device>
          </destination>
        </forward>
      </remove_forward>
    </vzadevm>
  </data>
</packet>
```

resize_drive

This call is available on Windows only.

Summary:

Resizes an EFD filesystem image mounted inside a Container.

Request specification:

Name	Min/Max	Type	Description
`resize_drive`			
`{`			
`size`	`1..1`	`long`	The new size in bytes. The minimum allowed size is 6 megabytes. There's no limit on the max size.

eid	0..1	eid_type (p. 25)	The ID of the server where the image is mounted.
[			
point	1..1	string	Mount point.
device	1..1	string	*This parameter is not currently used.*
]			
}			

Returns:

OK/Error

Errors

See devm Errors

Description:

For more info on creating and mounting EFD filesystem images, see the create_drive call (p. 163).

Example:

```
<packet version="4.0.0">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <resize_drive>
        <size>8000000</size>
        <eid>7f29d970-3e31-46f3-9b59-2654329e3e55</eid>
        <point>F:</point>
      </resize_drive>
    </vzadevm>
  </data>
</packet>
```

umount

Summary:

Unmounts a device that was previously mounted using the new_mount call (p. 180).

Request specification:

Name	Min/Max	Type	Description
umount		mount_deviceType (p. 160)	
{			
eid	1..1	eid_type (p. 25)	Server ID.
}			

Returns:

OK/Error

Description:

On Linux, the `umount` call is used to deactivate an existing mount point (i.e. to detach the specified filesystem from the file hierarchy), to change the mount point status from "permanent" to "temporary", or to remove the mount point from the system completely.

To deactivate the mount point, include the `point` parameter containing the directory name, the `eid` parameter containing the Server ID, and the `active` parameter containing the `true` value (see code examples below).

To change the status of the mount point from "permanent" to "temporary", include the `point` parameter containing the directory name, the `eid` parameter containing the Server ID, and the `permanent` parameter containing the `true` value (see code examples below). Please note that temporary mounts can only exist in the "active" state, therefore you cannot make a mount point temporary if it is not currently active.

To remove the mount point, include the `point` parameter containing the directory name, the `eid` parameter containing the Server ID, the `point` parameter containing the directory name, and the `permanent` parameter containing the `true` value (see code examples below).

On Windows, the `umount` call removes the drive that has been associated with a physical storage device on the Hardware Node or an EFD image. When unmounting an EFD image, the image file will not be physically deleted and can be re-mounted later. To permanently delete the image file, use the `delete_drive` call (p. 167). The following parameters are used:

`point` -- the drive letter (e.g. `E:`) associated with the device or the image file.

`eid` -- Server ID.

Example 1:

Deactivating the existing mount point.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <umount>
        <device>/dev/sda2</device>
        <point>/mydrive</point>
        <active>true</active>
        <eid>107d1f60-841e-8c43-8152-3c368ef3c366</eid>
      </umount>
    </vzadevm>
  </data>
</packet>
```

Example 2:

Making the mount point temporary.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <umount>
        <permanent>1</permanent>
        <point>/mydrive</point>
        <eid>107d1f60-841e-8c43-8152-3c368ef3c366</eid>
      </umount>
    </vzadevm>
  </data>
</packet>
```

Example 3:

Removing the mount point.

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <umount>
        <permanent>true</permanent>
        <point>/mydrive</point>
        <eid>107d1f60-841e-8c43-8152-3c368ef3c366</eid>
        <active>1</active>
      </umount>
    </vzadevm>
  </data>
</packet>
```

Example 4:

Unmounting a device that was previously mounted inside a Container (the device can be a physical drive or an EFD image).

```
<packet version="4.0.0" id="2">
  <target>vzadevm</target>
  <data>
    <vzadevm>
      <umount>
        <point>K:</point>
        <eid>b85f10fc-e42b-4e1c-a18b-85c6a25501b8</eid>
      </umount>
    </vzadevm>
  </data>
</packet>
```

env_samplem

Purpose:

Sample configuration management interface. Sample configurations are used to create Virtuozzo Container optimized for a particular purpose. For example, a general purpose Container may not require as much resources as a Container that will be hosting a database server, so instead of configuring each server individually, a sample configuration is created for each purpose in advance and is saved in a file on the Hardware Node. The information stored in these files can then be used to create new Containers. The `env_samplem` interface allows to create, modify, retrieve, and delete the sample configuration data.

Calls

Call	Description
get_sample_conf (p. 189)	Retrieves a list of sample configurations from the Hardware Node.
set_sample_conf	Modifies an existing sample configuration or creates a new one.
del_sample_conf	Deletes an existing sample configuration.

get_sample_conf

Summary:

Retrieves a list of sample configurations from a physical server.

Request specification:

Name	Min/Max	Type	Description
`get_sample_conf`			
`{`			
`id`	`0..[]`	`string`	A list of IDs of sample configurations to include in the result set. If none specified, all available sample configurations will be retrieved.
`eid`	`0..1`	`eid_type` (p. 25)	Server ID. If specified, retrieves only the sample configurations that can be used to create virtual machines on the specified host.
`}`			

Returns:

Name	Min/Max	Type	Description
`sample_conf`	`0..[]`	`sample_confType`	Sample configuration data.

Description:

The `get_sample_conf` call allows performing the following tasks:

- Retrieve a list of sample configurations available on the current server. If the server is a Master Node in a Virtuozzo group, the call retrieves sample configurations from every Node in the group. To perform this task, do not include any parameters.
- Retrieve a specific sample configuration. To perform this task, specify the sample configuration ID using the `id` element.
- Retrieve a list of sample configurations that can be used to create virtual servers on the specified hosts. Consider the following example. Let's say that you have a Virtuozzo group set up and you would like to create Containers on one of the Nodes in the group. Each Node in the group may contain unique sample configuration files, any of which can be used to create Virtuozzo Containers on any of the Nodes in the group. However, not all sample configurations may be compatible with the given Hardware Node because each sample configuration is designed for a specific platform (Linux, Windows), CPU architecture, etc. To get the list of the sample configurations that can be used to create Containers on a specific node, specify the Server ID of the Node using the `eid` parameter.

The following examples demonstrate how to perform each of the tasks described above.

Example 1:

Retrieving a list of all sample configurations available on the Hardware Node (or in the entire Virtuozzo group, if we are connected to the Master Node).

Input

```
<packet version="4.0.0">
  <target>env_samplem</target>
  <data>
    <env_samplem>
      <get_sample_conf/>
    </env_samplem>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/env_samplem"
xmlns:ns3="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="2c45ec0951t18ber1a0"
time="2007-03-05T09:09:34+0000" priority="0" version="4.0.0">
<ns1:origin>env_samplem</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:env_samplem>
    <ns2:sample_conf>
      <!-- Virtuozzo Container configuration data -->
      <ns2:env_config xsi:type="ns3:venv_configType">
        <ns4:os xsi:type="ns4:osType">
          <ns4:platform>Linux</ns4:platform>
          <ns4:name/>
```

```
        </ns4:os>
        <ns4:architecture>i686</ns4:architecture>
        <ns4:type>virtuozzo</ns4:type>
        <ns4:on_boot>0</ns4:on_boot>
        <ns4:offline_management>1</ns4:offline_management>
        <ns4:qos>
          <ns4:id>avnumproc</ns4:id>
          <ns4:hard>40</ns4:hard>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>cpuunits</ns4:id>
          <ns4:hard>1000</ns4:hard>
        </ns4:qos>
        <!-- The rest of the <qos> parameters are omitted for brevity -->
        <ns4:qos>
          <ns4:id>vmguarpages</ns4:id>
          <ns4:hard>2147483647</ns4:hard>
          <ns4:soft>6144</ns4:soft>
        </ns4:qos>
      </ns2:env_config>
      <!-- Sample configuration ID, name, etc. -->
      <ns2:id>0bd2ca65-8928-4f0d-8396-e8cba58dada0</ns2:id>
      <ns2:name>vps.confixx</ns2:name>
      <ns2:vt_version>
        <ns2:platform>Linux</ns2:platform>
        <ns2:architecture>i686</ns2:architecture>
        <ns2:vt_technology>virtuozzo</ns2:vt_technology>
      </ns2:vt_version>
    </ns2:sample_conf>
  </ns2:env_samplem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 2:

Retrieving a specific sample configuration.

Input

```
<packet version="4.0.0">
  <target>env_samplem</target>
  <data>
    <env_samplem>
      <get_sample_conf>
        <id>0bd2ca65-8928-4f0d-8396-e8cba58dada0</id>
       </get_sample_conf>
    </env_samplem>
  </data>
</packet>
```

The output here is similar to the output shown in the previous example.

Example 3:

Retrieving a list of sample configurations that can be used to create Virtuozzo Containers.

Input

```
<packet version="4.0.0" id="2">
  <target>server_group</target>
  <data>
    <server_group>
```

```
      <get_list>
        <type>generic</type>
      </get_list>
    </server_group>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="dc45ed4e57t41bbr274"
time="2007-03-06T09:05:31+0000" priority="0" version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient5</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns2:eid>565b96bd-d2da-4c7e-a212-0943a4bd6b29</ns2:eid>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Now, that we have the Server ID of the host, use the `get_sample_conf` call to get the list of the compatible sample configurations.

Input

```
<packet version="4.0.0">
  <target>env_samplem</target>
  <data>
    <env_samplem>
      <get_sample_conf>
        <eid>565b96bd-d2da-4c7e-a212-0943a4bd6b29</eid>
      </get_sample_conf>
    </env_samplem>
  </data>
</packet>
```

envm

Purpose:

The base server management interface that provides calls for physical and virtual server management. Supported virtualization technologies have their own server management interfaces, which are derived from the `envm` interface.

Derived interfaces:

`vzaenvm (p. 580)`

`vzpenvm` (p. 642)

Calls

Call	Description
create (p. 195)	Creates a new virtual server.
repair (p. 215)	Creates a Virtuozzo Container as a temporary replacement for another Container that needs repairs.
stop_repair (p. 226)	Stops and destroys the temporary Container created by the `repair` call.
start (p. 224)	Starts the specified Container.
stop (p. 225)	Stops the specified Container.
restart (p. 215)	Restarts a Container.
destroy (p. 201)	Destroys a Container.
suspend (p. 227)	Suspends a Container.
resume (p. 216)	Resumes a Container that was previously suspended with the `suspend` call.
get_info (p. 202)	Retrieves Container information.
get_list (p. 208)	Gets a list of Containers from a Hardware Node.
set (p. 217)	Sets the Container configuration parameters.
put_private	Creates or replaces a file in the Container private area.
get_private	Retrieves the contents of a file from the private area of the specified Container.
get_vt_settings (p. 212)	Retrieves Virtuozzo Containers settings.
set_vt_settings (p. 223)	Allows to modify Virtuozzo Containers settings.
get_vt_info (p. 211)	Retrieves read-only information about the Virtuozzo Containers software installed on the Hardware Node.
get_log (p. 210)	Retrieves Virtuozzo Containers logs.
get_native_config (p. 228)	Obtains a native Virtuozzo Container configuration based on the provided virtual configuration.
get_virtual_config	Obtains virtual configuration based on the provided native Container configuration.

create

Summary:

Creates a new virtual server.

This is a logged operation.

Request specification:

Name	Min/Max	Type	Description
create			
{			
force	0..1	none	Ignore pool problems and forcibly create a virtual server.
config	1..1	env_configType	Container configuration parameters. ▪ When creating a Virtuozzo Container, use venv_configType (p. 574), which is a Virtuozzo implementation of the config structure. ▪ When creating a virtual machine, use a VM specific venv_configType (p. 636).
default	0..1		A list of configuration parameters that should be set to default values. Use this option when you are using a sample configuration file but would like to use the default values for some of the parameters. If you are specifying all of the parameters manually, you can also use this list to set some of the parameters to defaults.
{			
parameter	1..[]	string	The names of the configuration parameter to set to default values.
}			
}			

Returns:

Name	Min/Max	Type	Description
env	0..[]	envType (p. 35)	The new virtual server information.

Description:

To create new Virtuozzo Containers, use `vzaenvm` (p. 580), which is a Parallels Virtuozzo implementation of this interface. The configuration parameters (the `config` element) must also be passed using the `venv_configType` (p. 574), which is a Virtuozzo implementation of the generic `env_configType` structure.

The Container configuration parameters can be passed explicitly by specifying the parameters and values or they can be passed by specifying the ID of a sample configuration file. Using a sample configuration file is a standard way of creating a Container. All of the parameters in the `venv_configType` (p. 574) structure are optional. You can pass just the sample configuration ID and that might be enough to create a Container if the configuration file contains all the necessary parameters. Some of those mandatory parameters are the OS template name, QoS counters, and some others. Some configuration parameters can only be set manually. For example, parameters like computer name, hostname, IP addresses will never have default values in a sample configuration file, so if you want to set them, you have to populate the appropriate elements of the configuration structure manually.

To create new virtual machines, use vzpenvm (p. 642), which is Parallels Server implementation of this interface. The configuration parameters for a new virtual machine must also be passed using the VM specific `venv_configType`.

Example 1:

Creating a Virtuozzo Container using the following parameters:

Parameter	Value	Description
`base_sample_id`	`9fb463e4-6f19-441c-9c5d-7dc26585b742`	The sample configuration ID. To retrieve a list of sample configurations from the Hardware Node, use `get_sample_conf`. (p. 189)
`os_template/name`	`redhat-as3-minimal`	OS template name. To retrieve the list of the available OS template, use `packagem/list` (p. 337).
`name`	`Test-VE5`	Computer name.
`hostname`	`Host-105`	Hostname.
`veid`	`105`	Container ID.
`on_boot`	`true`	Start the Container automatically on system boot.
`offline_management`	`true`	Turn the offline-management feature on for the Container.

ip_address	10.17.3.125	IP address. We will assign the address to the default venet0 virtual network adapter. The venet0 adapter is created automatically for every Container. We could also create our own virtual network adapter inside a Container and customize it according to our needs. For more info on how to create and configure virtual ethernet adapters, see venv_configType (p. 574) and net_vethType (p. 569).

Input:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <create>
        <config >
          <name>Test-VE5</name>
          <hostname>Host-105</hostname>
          <base_sample_id>e89373d0-d13c-1741-a0e5-
212d7cd3ae61</base_sample_id>
          <veid>106</veid>
          <on_boot>true</on_boot>
          <offline_management>true</offline_management>
          <os_template>
            <name>redhat-as3-minimal</name>
          </os_template>
          <net_device>
            <id>venet0</id>
            <ip_address>
              <ip>10.17.3.125</ip>
            </ip_address>
            <host_routed/>
          </net_device>
        </config>
      </create>
    </vzaenvm>
  </data>
</packet>
```

Output:

The output contains the new Container information, including the Server ID (EID).

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/envm"
xmlns:ns3="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="fc468cd97at5f90redc"
time="2007-07-05T14:33:42+0000" priority="4000" version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:target>vzclient3-cfa5a2f6-4bc8-9140-88a2-15b1eae98cac</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:env xsi:type="ns3:envType">
      <ns4:parent_eid>00000000-0000-0000-0000-000000000000</ns4:parent_eid>
      <ns4:eid>78c510e9-6e4f-5349-9e22-48841e709fea</ns4:eid>
      <ns4:status xsi:type="ns4:env_statusType">
        <ns4:state>1</ns4:state>
      </ns4:status>
      <ns4:alert>0</ns4:alert>
      <ns4:config xsi:type="ns4:env_configType"/>
      <ns4:virtual_config xsi:type="ns3:venv_configType">
        <base_sample_id>e89373d0-d13c-1741-a0e5-212d7cd3ae61</base_sample_id>
        <veid>106</veid>
        <on_boot>true</on_boot>
        <offline_management>true</offline_management>
        <os_template>
          <name>redhat-as3-minimal</name>
        </os_template>
        <net_device>
          <id>venet0</id>
```

```
          <ip_address>
            <ip>10.17.3.125</ip>
          </ip_address>
          <host_routed/>
        </net_device>
        <ns4:architecture>i386</ns4:architecture>
        <ns4:address>
          <ns4:ip>0.0.0.0</ns4:ip>
        </ns4:address>
        <ns4:qos>
          <ns4:id>avnumproc</ns4:id>
          <ns4:hard>40</ns4:hard>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>cpuunits</ns4:id>
          <ns4:hard>1000</ns4:hard>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>dcachesize</ns4:id>
          <ns4:hard>1097728</ns4:hard>
          <ns4:soft>1048576</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>dgramrcvbuf</ns4:id>
          <ns4:hard>132096</ns4:hard>
          <ns4:soft>132096</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>diskinodes</ns4:id>
          <ns4:hard>220000</ns4:hard>
          <ns4:soft>200000</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>diskspace</ns4:id>
          <ns4:hard>1153434</ns4:hard>
          <ns4:soft>1048576</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>kmemsize</ns4:id>
          <ns4:hard>2936012</ns4:hard>
          <ns4:soft>2752512</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>lockedpages</ns4:id>
          <ns4:hard>32</ns4:hard>
          <ns4:soft>32</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>numfile</ns4:id>
          <ns4:hard>2048</ns4:hard>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>numflock</ns4:id>
          <ns4:hard>110</ns4:hard>
          <ns4:soft>100</ns4:soft>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>numiptent</ns4:id>
          <ns4:hard>128</ns4:hard>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>numothersock</ns4:id>
          <ns4:hard>80</ns4:hard>
        </ns4:qos>
        <ns4:qos>
          <ns4:id>numproc</ns4:id>
          <ns4:hard>65</ns4:hard>
```

```
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numpty</ns4:id>
        <ns4:hard>16</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numsiginfo</ns4:id>
        <ns4:hard>256</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numtcpsock</ns4:id>
        <ns4:hard>80</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>oomguarpages</ns4:id>
        <ns4:hard>2147483647</ns4:hard>
        <ns4:soft>6144</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>othersockbuf</ns4:id>
        <ns4:hard>336896</ns4:hard>
        <ns4:soft>132096</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>physpages</ns4:id>
        <ns4:hard>2147483647</ns4:hard>
        <ns4:soft>0</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>privvmpages</ns4:id>
        <ns4:hard>24576</ns4:hard>
        <ns4:soft>22528</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>quotatime</ns4:id>
        <ns4:hard>0</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>shmpages</ns4:id>
        <ns4:hard>8192</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>slmmemorylimit</ns4:id>
        <ns4:hard>33521664</ns4:hard>
        <ns4:soft>33521664</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>tcprcvbuf</ns4:id>
        <ns4:hard>524288</ns4:hard>
        <ns4:soft>319488</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>tcpsndbuf</ns4:id>
        <ns4:hard>524288</ns4:hard>
        <ns4:soft>319488</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>vmguarpages</ns4:id>
        <ns4:hard>2147483647</ns4:hard>
        <ns4:soft>6144</ns4:soft>
      </ns4:qos>
      <ns3:origin_sample>basic</ns3:origin_sample>
      <ns4:name>Test-VE5</ns4:name>
      <ns4:hostname>Host-105</ns4:hostname>
    </ns4:virtual_config>
  </ns2:env>
</ns2:vzaenvm>
```

```
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 2:

The following example creates a Container on a Windows platform. As in the Linux example above, we also use a sample configuration file and setting some of the parameters manually, including computer name, hostname, Container ID (veid), the "on-boot" parameter, and the IP address for the default `venet0` network adapter.

Input

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <create>
        <config >
          <name>Test-VE5</name>
          <hostname>Host-105</hostname>
          <base_sample_id>46413905-b2d7-4f41-bcd6-
e2662e63cd63</base_sample_id>
          <veid>105</veid>
          <on_boot>true</on_boot>
          <net_device>
            <id>venet0</id>
            <ip_address>
              <ip>10.17.3.125</ip>
            </ip_address>
            <host_routed/>
          </net_device>
        </config>
      </create>
    </vzaenvm>
  </data>
</packet>
```

destroy

Summary:

Destroys a virtual server.

A logged operation.

Request specification:

Name	Min/Max	Type	Description
`destroy`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	Server ID.
`}`			

Returns:

OK/Error.

Description:

The call destroys the specified virtual server. All the virtual server data is removed from the physical server and cannot be recovered. You can only destroy a stopped virtual server.

To destroy a Virtuozzo Container, use the vzaenvm (p. 580) interface.

To destroy a virtual machine, use the vzpenvm (p. 642) interface.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <destroy>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
      </destroy>
    </vzaenvm>
  </data>
</packet>
```

get_info

Summary:

Retrieves a Hardware Node or a Virtuozzo Container, or a virtual machine configuration information.

Request specification:

Name	Min/Max	Type	Description
get_info			
{			
eid	0..[]	eid_type (p. 25)	Server ID. When retrieving information for a Hardware Node, this element may be omitted. When retrieving information for a Virtuozzo Container, specify its Server ID, or omit the element to retrieve information for all Containers on the current host.
config	0..1	none	Include this element to retrieve the Container or virtual machine configuration information. If the element is omitted, the output will contain only the basic information.

filter_config	0..1		Specify the configuration parameters that you want to be included in the output. The config element (above) must also be included in the request. If this element is omitted then all available configuration parameters will be retrieved.
{			
<xs:any>		xs:any	A list containing the names of the configuration parameter to include in the output. For a list of the available parameters see env_configType structure (Hardware Node information), or venv_configType (p. 574) (Virtuozzo Container configuration), or VM specific venv_configType (p. 636) (.virtual machine configuration)
}			
}			

Returns:

Name	Min/Max	Type	Description
env	0..[]	envType (p. 35)	Configuration information.

Description:

The call is available in the base envm interface (p. 193) and in its descendants, the vzaenvm interface (p. 580) (for Virtuozzo Containers) and vzpenvm interface (p. 642) (for virtual machines). Use the envm interface to retrieve the information for the Hardware Node that you are currently connected to. Use the vzaenvm interface to get the information for a particular Virtuozzo Container or vzpenvm interface to get the information for a particular virtual machine.

When retrieving information for a Virtuozzo Container, the output will contain the Container virtual configuration information (p. 574). Most of the configuration parameters are optional, so some may not be included in the output structure. If a parameter is not included, it means that its default value is currently used. To determine the default value, first use the envm/get_vt_setting (p. 212) call. If you don't see the parameter in the output, then, depending on the parameter data type, the default value is determined as follows:

Data Type	Default Value
boolean	false
string	Empty string

int	0 or maxint

Example:

Retrieving information for the Hardware Node.

Input

```
<packet version="4.0.0" id="2">
  <target>envm</target>
  <data>
    <envm>
      <get_info>
        <config/>
      </get_info>
    </envm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/envm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" time="2007-11-
21T07:10:35+0000" id="8c4747000ct18ber448" priority="0" version="4.0.0">
<origin>envm</origin>
<target>vzclient6-a0f0a8d0-6c35-c64b-8f16-c1f0e13295c6</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <envm>
    <env xsi:type="ns2:envType">
      <parent_eid>00000000-0000-0000-0000-000000000000</parent_eid>
      <eid>a0f0a8d0-6c35-c64b-8f16-c1f0e13295c6</eid>
      <status xsi:type="ns2:env_statusType">
        <state>6</state>
      </status>
      <alert>0</alert>
      <config xsi:type="ns2:env_configType">
        <type>generic</type>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <kernel>2.6.18-028stab049.1</kernel>
          <name>Red Hat Enterprise Linux Server release 5 (Tikanga)</name>
        </os>
        <architecture>i386</architecture>
        <hostname>dhcp-10-30-22-205.sw.ru</hostname>
        <name>dhcp-10-30-22-205.sw.ru</name>
        <address>
          <ip>10.30.22.205</ip>
        </address>
        <nameserver>10.30.0.1</nameserver>
      </config>
      <virtual_config xsi:type="ns2:venv_configType"/>
    </env>
  </envm>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

Example:

Retrieving information for a Virtuozzo Container.

Input

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_info>
        <eid>a5961178-14d2-40cc-b1e7-41b562a2f4c6</eid>
        <config/>
      </get_info>
    </vzaenvm>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:target>vzclient5</ns1:target>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:env xsi:type="ns3:envType">
      <ns3:parent_eid>89e27960-97b8-461f-902f-557b4b16784b</ns3:parent_eid>
      <ns3:eid>3e25fee2-1163-4336-9e74-8b8097936d47</ns3:eid>
      <ns3:status xsi:type="ns3:env_statusType">
        <ns3:state>6</ns3:state>
      </ns3:status>
      <ns3:alert>0</ns3:alert>
      <ns3:config xsi:type="ns3:env_configType"/>
      <ns3:virtual_config xsi:type="ns4:venv_configType">
        <ns3:hostname>myhost</ns3:hostname>
        <ns3:name>Mycomputer</ns3:name>
        <ns3:offline_management>1</ns3:offline_management>
        <ns3:on_boot>1</ns3:on_boot>
        <ns3:os_template>
          <ns3:version>20061020</ns3:version>
          <ns3:name>redhat-as3-minimal</ns3:name>
        </ns3:os_template>
        <ns3:ve_root>/vz/root/$VEID</ns3:ve_root>
        <ns3:ve_private>/vz/private/$VEID</ns3:ve_private>
        <ns3:ve_type>
          <ns3:veid>0</ns3:veid>
          <ns3:type>1</ns3:type>
        </ns3:ve_type>
        <ns3:qos>
          <ns3:id>avnumproc</ns3:id>
          <ns3:hard>40</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>cpuunits</ns3:id>
          <ns3:hard>1000</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>dcachesize</ns3:id>
          <ns3:hard>1097728</ns3:hard>
          <ns3:soft>1048576</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>dgramrcvbuf</ns3:id>
          <ns3:hard>132096</ns3:hard>
          <ns3:soft>132096</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>diskinodes</ns3:id>
          <ns3:hard>220000</ns3:hard>
          <ns3:soft>200000</ns3:soft>
        </ns3:qos>
```

```
        <ns3:qos>
          <ns3:id>diskspace</ns3:id>
          <ns3:hard>1153434</ns3:hard>
          <ns3:soft>1048576</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>kmemsize</ns3:id>
          <ns3:hard>2936012</ns3:hard>
          <ns3:soft>2752512</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>lockedpages</ns3:id>
          <ns3:hard>32</ns3:hard>
          <ns3:soft>32</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numfile</ns3:id>
          <ns3:hard>2048</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numflock</ns3:id>
          <ns3:hard>110</ns3:hard>
          <ns3:soft>100</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numiptent</ns3:id>
          <ns3:hard>128</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numothersock</ns3:id>
          <ns3:hard>80</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numproc</ns3:id>
          <ns3:hard>65</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numpty</ns3:id>
          <ns3:hard>16</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numsiginfo</ns3:id>
          <ns3:hard>256</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numtcpsock</ns3:id>
          <ns3:hard>80</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>oomguarpages</ns3:id>
          <ns3:hard>2147483647</ns3:hard>
          <ns3:soft>6144</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>othersockbuf</ns3:id>
          <ns3:hard>336896</ns3:hard>
          <ns3:soft>132096</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>physpages</ns3:id>
          <ns3:hard>2147483647</ns3:hard>
          <ns3:soft>0</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>privvmpages</ns3:id>
          <ns3:hard>24576</ns3:hard>
          <ns3:soft>22528</ns3:soft>
```

```
        </ns3:qos>
        <ns3:qos>
          <ns3:id>quotatime</ns3:id>
          <ns3:hard>0</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>shmpages</ns3:id>
          <ns3:hard>8192</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>tcprcvbuf</ns3:id>
          <ns3:hard>524288</ns3:hard>
          <ns3:soft>319488</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>tcpsndbuf</ns3:id>
          <ns3:hard>524288</ns3:hard>
          <ns3:soft>319488</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>vmguarpages</ns3:id>
          <ns3:hard>2147483647</ns3:hard>
          <ns3:soft>6144</ns3:soft>
        </ns3:qos>
        <ns3:veid>101</ns3:veid>
        <ns3:type>virtuozzo</ns3:type>
        <ns3:offline_service>vzpp</ns3:offline_service>
        <ns3:offline_service>vzpp-plesk</ns3:offline_service>
        <ns3:os xsi:type="ns3:osType">
          <ns3:platform>Linux</ns3:platform>
          <ns3:kernel>2.6.9-023stab033.6</ns3:kernel>
          <ns3:version>20061020</ns3:version>
          <ns3:name>redhat-as3-minimal</ns3:name>
        </ns3:os>
        <ns3:net_device xsi:type="ns4:net_vethType">
          <ns3:id>venet0</ns3:id>
          <ns3:ip_address>
            <ns3:ip>10.100.23.203</ns3:ip>
          </ns3:ip_address>
          <ns4:host_routed/>
        </ns3:net_device>
        <ns3:address>
          <ns3:ip>10.100.23.203</ns3:ip>
        </ns3:address>
      </ns3:virtual_config>
    </ns2:env>
  </ns2:vzaenvm>
</ns1:data>
</ns1:packet>
```

get_list

Summary:

Retrieves Server IDs of the Hardware Node and Virtuozzo Containers or virtual machines that it hosts.

Request specification:

Name	Min/Max	Type	Description
`get_list`			
`{`			

count	0..1	int	The maximum number of the IDs to include in the list.
type	0..1	string	Retrieve only the servers of the specified type. *The parameter is not currently used.*
status	0..[]	env_statusType (p. 34)	Retrieve only the Containers or virtual machines, which status match one of the statuses specified here. By using this parameter, you can, for example, retrieve only the IDs of the running or stopped Containers or virtual machines, etc.
}			

Returns:

Name	Min/Max	Type	Description
eid	0..[]	eid_type (p. 25)	A list of Server IDs.

Description:

To retrieve a list of Virtuozzo Containers from the Hardware Node, use `vzaenvm` (p. 580), which is a Virtuozzo implementation of this interface.

To retrieve a list of virtual machines from the Hardware Node, use `vzpenvm` (p. 642), which is Parallels Server implementation of this interface.

Example:

Retrieving a list of running Virtuozzo Containers.

Input

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_list>
        <status>
          <state>6</state>
        </status>
      </get_list>
    </vzaenvm>
  </data>
</packet>
```

Output

```
<ns1:packet version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:eid>3e25fee2-1163-4336-9e74-8b8097936d47</ns2:eid>
    <ns2:eid>72145bf0-7562-43d4-b707-cc33d37e3f10</ns2:eid>
    <ns2:eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</ns2:eid>
  </ns2:vzaenvm>
</ns1:packet>
```

get_log

Summary:

Retrieves Virtuozzo Containers log data.

Request specification:

Name	Min/Max	Type	Description
`get_log`			
`{`			
`start_time`	`0..1`	`datetime_type` (p. 25)	Start time of the log.
`end_time`	`0..1`	`datetime_type` (p. 25)	End time of the log.
`records`	`0..1`	`int`	The number of records from the end of the log to include in the result set. If this element is omitted, all available records will be retrieved.

options	0..1	log_optionsType (p. 568)	Additional options.
}			

Returns:

Name	Min/Max	Type	Description
log	0..1	base64Binary	Log data.

Example:

Input

```
<packet version="4.0.0">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_log>
        <start_time>2006-05-28T19:05:30-0500</start_time>
        <end_time>2007-08-28T19:05:30-0500</end_time>
        <records>10</records>
        <options>
          <type>1</type>
        </options>
      </get_log>
    </vzaenvm>
  </data>
</packet>
```

get_vt_info

Summary:

Retrieves the read-only Virtuozzo Containers information.

Request specification:

Name	Min/Max	Type	Description
get_vt_info	1..1	none	

Returns:

Name	Min/Max	Type	Description
vt_info	0..1	vt_infoType (p. 63)	

Example:

Using the Virtuozzo implementation of the interface to retrieve Virtuozzo information.

Input

```
<packet>
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_vt_info/>
    </vzaenvm>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:vt_info xsi:type="ns3:vt_infoType">
      <ns3:version>4.0.0-112.swsoft</ns3:version>
      <ns3:release>?</ns3:release>
      <ns3:sve_eid>72145bf0-7562-43d4-b707-cc33d37e3f10</ns3:sve_eid>
      <ns3:hwid>BD95.2511.D49F.422B.6247.EAFC.0423.E8C2</ns3:hwid>
    </ns2:vt_info>
  </ns2:vzaenvm>
</ns1:data>
</ns1:packet>
```

get_vt_settings

Summary:

Retrieves virtualization technology specific settings.

Request specification:

Name	Min/Max	Type	Description
get_vt_settings	1..1	none	To get Virtuozzo Containers specific settings, use vzaenvm (p. 580), which is a Virtuozzo implementation of this interface. To get virtual machines specific settings, use vzpenvm (p. 642), which is Parallels Server implementation of this interface.

Returns:

`vt_settings` or `Error`.

Name	Min/Max	Type	Description
`vt_settings`	`0..1`	`vt_settingsType (p. 63)`	The actual type returned is: ▪ For Virtuozzo Containers:`vt_settingsType` (p. 578), the Virtuozzo Containers implementation of the generic `vt_settingsType`. ▪ For virtual machines: VM-specific vt_settingsType on page 640.

Example. Getting Virtuozzo Containers specific settings:

Input

```
<packet>
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_vt_settings/>
    </vzaenvm>
  </data>
</packet>
```

Output

```
<ns1:packet version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:vt_settings xsi:type="ns3:vt_settingsType">
      <ns3:lock_dir>L3Z6L2xvY2s=</ns3:lock_dir>
      <ns3:template_dir>L3Z6L3RlbXBsYXRl</ns3:template_dir>
      <ns3:parameter>
        <ns3:id>actionlogdir</ns3:id>
        <ns3:value>/vz/actionlog</ns3:value>
      </ns3:parameter>
      <ns3:parameter>
        <ns3:id>backups_mode</ns3:id>
        <ns3:value>standard</ns3:value>
      </ns3:parameter>
      <ns3:parameter>
        <ns3:id>bandwidth</ns3:id>
        <ns3:value>eth0:102400</ns3:value>
      </ns3:parameter>
      <ns3:parameter>
        <ns3:id>def_ostemplate</ns3:id>
        <ns3:value>fedora-core-4</ns3:value>
      </ns3:parameter>
      <ns3:parameter>
        <ns3:id>disk_quota</ns3:id>
        <ns3:value>yes</ns3:value>
      </ns3:parameter>
      <!-- Some of the output is omitted for brevity -->
      <ns3:offline_service xsi:type="ns3:redirect_serviceType">
        <ns3:id>vzpp-plesk</ns3:id>
        <ns3:port>8443</ns3:port>
        <ns3:dst>72145bf0-7562-43d4-b707-cc33d37e3f10</ns3:dst>
        <ns3:default/>
      </ns3:offline_service>
      <ns3:qoses/>
    </ns2:vt_settings>
  </ns2:vzaenvm>
</ns1:data>
</ns1:packet>
```

repair

Summary:

Creates a Virtuozzo Container as a temporary replacement for another Container that needs repair.

Request specification:

Name	Min/Max	Type	Description
`repair`			
{			
`eid`	`1..1`	`eid_type` (p. 25)	The Server ID of the original Container.
}			

Returns:

OK/Error.

Description:

If you have a Virtuozzo Container that needs repair or maintenance, you may use this call to create a new Container that will act as a temporary replacement for your original Container for the duration of the repairs. The call will create an exact copy of the specified Container, start it, and will stop the original Container, all with zero-downtime, so the user will be able to continue using the Container without interruption. Once you are done repairing the original Container, use the `stop_repair` call (p. 226) to revert to it.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <repair>
        <eid>ba17a0c5-9036-473c-a813-aa6f5b36cf16</eid>
      </repair>
    </vzaenvm>
  </data>
</packet>
```

restart

Summary:

Restarts the specified virtual server.

A logged operation.

Request specification:

Name	Min/Max	Type	Description
`restart`			

{			
eid	1..1	eid_type (p. 25)	The Server ID.
}			

Returns:

OK/Error.

Description:

The call stops and then automatically starts the specified virtual server. If the virtual server is not currently running, the call skips the stopping part and simply starts the virtual server.

To restart a Virtuozzo Container, use the vzaenvm (p. 580) interface.

To restart a virtual machine, use the vzpenvm (p. 642) interface.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <restart>
        <eid>ba17a0c5-9036-473c-a813-aa6f5b36cf16</eid>
      </restart>
    </vzaenvm>
  </data>
</packet>
```

resume

Summary:

Resumes a Virtuozzo Container that was previously suspended by the suspend call.

Request specification:

Name	Min/Max	Type	Description
resume			
{			
eid	1..1	eid_type (p. 25)	The Server ID of the Container to resume.
}			

Returns:

OK/Error

Description:

For the Virtuozzo-specific implementation of this call see the `vzaenvm` interface (p. 580).

To resume a virtual machine, use the vzpenvm (p. 642) interface.

set

Summary:

Sets the server configuration parameters.

Note: for virtual machines, only `eid` and `config` are passed to the `set` call, see **Request specification (specifying parameters manually)** below. Other options are applicable to Virtuozzo Containers only.

Request specification (specifying parameter and values manually):

Name	Min/Max	Type	Description
`set`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	Server ID.
`config`	`1..1`	`env_configType`	Server configuration. ▪ When modifying Virtuozzo Container configuration, use `venv_configType` (p. 574). ▪ When modifying a virtual machine, use a VM specific venv_configType (p. 636).
`force`	`0..1`	`none`	For Virtuozzo Containers only. Ignore possible pool problems and forcibly assign the IP address.
`default`	`0..1`		For Virtuozzo Containers only. Use this element to specify a list of configuration parameters for which you want to use the default values.

{			
parameter	1..[]	string	For Virtuozzo Containers only. Parameter name.
}			
set_mode	0..1	string	For Virtuozzo Containers only. Specifies the operation mode: restart -- restart the server if it is required to do so for a new parameter value to take effect. ignore -- ignore possible errors while applying the new values to the running server.

Note: Request specification below is applicable to Virtuozzo Containers only.

Request specification (using values from a sample configuration):

Name	Min/Max	Type	Description
set			
{			
eid	1..1	eid_type (p. 25)	Server ID.
apply_config	0..1		
{			
sample_conf	1..1	guid_type (p. 26)	Sample configuration ID. To obtain a list of the available sample configurations, use get_sample_conf (p. 189).

parameter	0..[]	string	A list of the parameters to set. See env_configType for the parameter names. ▪ When modifying Virtuozzo Container configuration, see venv_configType (p. 574). ▪ When modifying a virtual machine, use a VM specific venv_configType (p. 636). **Note:** The template and network specific parameters cannot be modified using this function. These parameters are: template, os_template, ve_root, ve_private, hostname, ip_address.
category	0..[]	string	A list of parameter categories. If you would like to set an entire parameter category (or multiple categories), specify it here. For the list of categories see Description below.
config_customized	0..1	boolean	A flag indicating that the server configuration has been customized after the server was created. Set this element to true to save the flag in the configuration file for future reference.
}			

set_mode	0..1	string	Specifies the operation mode: restart -- restart the server if it is required to do so for the new value to take effect. ignore -- ignore possible errors while applying the new values to the running server.
}			

Returns:

Name	Min/Max	Type	Description
env_config	0..1	env_configType	The updated configuration information.

Description:

Specifying parameter and values manually

Using the syntax #1 (above), you can pass a list of parameters (or a single parameter) with values that you would like to modify. The configuration parameters are specified using the config element. Although you can modify any configuration parameter that you like, you should only use this approach to set the parameters that cannot possibly break a Container (e.g. hostname, DNS servers, etc.). Speaking about Virtuozzo Container, when modifying QoS-related parameters, always make sure that you know exactly what you are doing, or use the second approach described below. For a sample XML request, see Example 1-3 below.

Using values from a sample configuration

This approach (syntax #2) also allows to specify the name of the parameters but their values will be taken from a sample configuration. This is useful when setting (or re-setting) the QoS values because a sample configuration contains the values that are fine-tuned for the type of application that you intend to run inside the Container. Although you can modify individual parameters, it often makes sense to modify an entire parameter category. This is accomplished by specifying the category ID using the category element. The following table lists the categories that can be set using this approach.

Category ID	Description
general_conf	General Container parameters.
qos	Resource parameters - UBC, disk quota, CPU - all at once.
quota	Disk quota parameters.
cpu	CPU parameters.

For a sample XML request, see Example 4 below.

When using either approach, the new values are applied to the server immediately and are saved in the configuration file, making the configuration changes permanent.

Example 1:

Assigning a new hostname and adding a search domain to a Virtuozzo Container. Also adding two DNS servers to the default `venet0` virtual network adapter.

Input

```
<packet version="4.0.0">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set>
        <eid>3288bb6b-8a49-4230-b565-6ad5521182aa</eid>
        <config>
          <hostname>myhost</hostname>
          <search_domain>ts6.com</search_domain>
          <net_device>
            <id>venet0</id>
            <nameserver>192.168.1.51</nameserver>
            <nameserver>192.168.1.52</nameserver>
          </net_device>
        </config>
      </set>
    </vzaenvm>
  </data>
</packet>
```

Example 2:

Modifying the IP address for `venet0` network adapter, which is the default virtual adapter inside a Virtuozzo Container. This modification works in such a way that the existing IP addresses are first removed from the adapter's configuration and then the new addresses are added. To add an IP address, first retrieve the existing addresses, then add the new address (or addresses) to the list, and then include the entire list in the request. On Linux, when modifying the default `venet0` adapter, the `host_routed` element must be present in the request. Configuring a non-default virtual network adapter is similar with one exception: you cannot use the `host_routed` mode, so you have to attach the adapter to an existing Virtuozzo virtual network by including the `network_id` element containing the ID of the virtual network.

Input

```
<packet version="4.0.0">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set>
        <eid>72145bf0-7562-43d4-b707-cc33d37e3f10</eid>
        <config>
          <net_device>
            <id>venet0</id>
            <ip_address>
              <ip>10.130.1.1</ip>
            </ip_address>
            <ip_address>
              <ip>10.130.1.2</ip>
```

```
            </ip_address>
            <ip_address>
              <ip>10.130.1.3</ip>
            </ip_address>
            <host_routed/>
          </net_device>
        </config>
      </set>
    </vzaenvm>
  </data>
</packet>
```

Example 3:

This example is a Windows version of the previous example (modifying the IP address configuration for the default `venet0` network adapter). The difference here is that you may or may not have to include the `host_routed` element depending on the following conditions:

- Include the element if you would like to set the adapter to use the `host_routed` mode.
- Don't include it if you want the adapter to use the `bridged` mode. When using the `bridged` mode, you must also specify the virtual network ID to connect the adapter to by populating the `network_id` field. See `net_vethType` (p. 569) for more information.
- Configuring any other (non-default) virtual network adapter is exactly the same as configuring the default `venet0` adapter.

In our example, we are switching the adapter to the `bridged` mode and attaching it to the specified Virtuozzo virtual network. For more information on Virtuozzo virtual networks, see the `vzanetworkm` interface (p. 606).

Input

```
<packet version="4.0.0">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set>
        <eid>72145bf0-7562-43d4-b707-cc33d37e3f10</eid>
        <config>
          <net_device>
            <id>venet0</id>
            <ip_address>
              <ip>10.130.1.1</ip>
            </ip_address>
            <ip_address>
              <ip>10.130.1.2</ip>
            </ip_address>
            <ip_address>
              <ip>10.130.1.3</ip>
            </ip_address>
            <network_id>dnpuZXQx</network_id>
          </net_device>
        </config>
      </set>
    </vzaenvm>
  </data>
</packet>
```

Example 4:

Setting an entire set of QoS parameters using the values from the specified sample configuration.

Input

```
<packet version="4.0.0" id="654">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
        <apply_config>
          <sample_conf>f8e96630-7fd8-4eee-93b2-3ad7b6b53916</sample_conf>
          <category>qos</category>
        </apply_config>
      </set>
    </vzaenvm>
  </data>
</packet>
```

set_vt_settings

Summary:

Sets virtualization technology specific settings.

Request specification:

Name	Min/Max	Type	Description
set_vt_settings			
{			
vt_settings	1..1	vt_settingsType (p. 63)	Virtuozzo Containers settings. Use vt_settingsType (p. 578), the Virtuozzo implementation of the generic vt_settingsType structure. For virtual machines use VM- specific vt_settingsType on page 640.
}			

Returns:

OK/Error.

Example:

Modifying the default OS template for the Virtuozzo Containers installation.

```
<packet>
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set_vt_settings>
        <vt_settings>
          <parameter>
            <id>def_ostemplate</id>
            <value>redhat-as3-minimal</value>
          </parameter>
        </vt_settings>
      </set_vt_settings>
    </vzaenvm>
  </data>
</packet>
```

start

Summary:

Starts the specified virtual server.

A logged operation.

Request specification:

Name	Min/Max	Type	Description
start			
{			
eid	1..1	eid_type (p. 25)	Server ID.
}			

Returns:

OK/Error.

Description:

The call starts the specified virtual server. If the virtual server cannot be started or is already running, the call will return an error.

To start a Virtuozzo Container, use the vzaenvm (p. 580) interface.

To start a virtual machine, use the vzpenvm (p. 642) interface.

Examples:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <start>
        <eid>ba17a0c5-9036-473c-a813-aa6f5b36cf16</eid>
      </start>
    </vzaenvm>
  </data>
</packet>
```

stop

Summary:

Stops the specified virtual server.

A logged operation.

Request specification:

Name	Min/Max	Type	Description
stop			
{			
eid	1..1	eid_type (p. 25)	Server ID.
force	0..1	none	Forcibly stop the virtual server.
}			

Returns:

OK/Error.

Description:

The call stops the specified virtual server. If the virtual server cannot be stopped, an error code will be returned. You can try to forcibly stop the virtual server by including the `force` element in the request.

To stop a Virtuozzo Container, use the vzaenvm (p. 580) interface.

To stop a virtual machine, use the vzpenvm (p. 642) interface.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <stop>
        <eid>ba17a0c5-9036-473c-a813-aa6f5b36cf16</eid>
      </stop>
    </vzaenvm>
  </data>
</packet>
```

stop_repair

Summary:

Stops and destroys the temporary Container created by the `repair` call (p. 215).

Request specification:

Name	Min/Max	Type	Description
`stop_repair`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	The Server ID of the original Container.
`}`			

Returns:

OK/Error.

Description:

This call will stop and destroy the temporary Container created by the `repair` call (p. 215). It will then bring the original Container back up. Execute this call after you are done repairing the original Container and want to revert to it.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <stop_repair>
        <eid>ba17a0c5-9036-473c-a813-aa6f5b36cf16</eid>
      </stop_repair>
    </vzaenvm>
  </data>
</packet>
```

suspend

Summary:

Suspends a virtual server.

Request specification:

Name	Min/Max	Type	Description
`suspend`			
`{`			
`  eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`}`			

Returns:

OK/Error

Description:

Use this call to temporarily suspend a virtual server without shutting it down. The status of a virtual server becomes "suspended". To resume a virtual server, use the `resume` call.

For the Virtuozzo-specific implementation of this call see the `vzaenvm` interface (p. 580).

To suspend a virtual machine, use the vzpenvm (p. 642) interface.

get_native_config

Summary:

Converts a Container configuration data from the Agent format to the Virtuozzo Containers native format.

Request specification:

Name	Min/Max	Type	Description
`get_native_config`			
`{`			
`virtual_config`	`1..1`	`venv_configType` (p. 61)	Container configuration data in the Agent format. To obtain the Container configuration from Agent, use the `get_info` call (p. 202).
`}`			

Returns:

Name	Min/Max	Type	Description
`native_config`	`0..[]`	`native_configType (p. 47)`	Virtual configuration data in the Virtuozzo Containers native format. The appropriate subtype of `native_configType (p. 47)` will be used in the result.

Description:

Parallels Agent uses its own data structures for the Virtuozzo Containers configuration data (the subtypes of `venv_configType` (p. 61)). You use this data structures when creating, examine, or modifying a Virtuozzo Container through Parallels Agent API. The Virtuozzo Containers software stores the same configuration data differently. It uses the bash style configuration data formatting, which is a set of values in the `VARNAME="value-string"` form. The `get_native_config` call allows to convert the Agent version of the configuration data to the native Virtuozzo Containers formatting.

Example:

In the following example, we pass the Agent version of the Container configuration data to the `get_native_config` call.

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_native_config>
         <virtual_config xsi:type="ns2:venv_configType">
           <on_boot>0</on_boot>
           <qos>
             <id>cpuunits</id>
             <hard>1000</hard>
           </qos>
           <qos>
             <id>dgramrcvbuf</id>
             <hard>133120</hard>
             <soft>133120</soft>
           </qos>
           <qos>
             <id>diskinodes</id>
             <hard>440000</hard>
           <soft>400000</soft>
           </qos>
           <qos>
             <id>diskspace</id>
             <hard>11141120</hard>
             <soft>10485760</soft>
           </qos>
           <qos>
             <id>kmemsize</id>
             <hard>17107200</hard>
             <soft>15582912</soft>
           </qos>

           <!-- The rest of the configuration parameters
                are omitted for brevity -->
         </virtual_config>
       </get_native_config>
     </vzaenvm>
   </data>
</packet>
```

Output

The Virtuozzo native configuration data is received as a block of base64-encoded data. After you decode it, the result will look similar to the following example:

```
VERSION="2"
ONBOOT="no"
AVNUMPROC="40:40"
NUMPROC="65:65"
NUMTCPSOCK="80:80"
NUMOTHERSOCK="80:80"
VMGUARPAGES="6144:2147483647"
KMEMSIZE="2752512:2936012"
TCPSNDBUF="319488:524288"
TCPRCVBUF="319488:524288"
OTHERSOCKBUF="132096:336896"
DGRAMRCVBUF="132096:132096"
OOMGUARPAGES="6144:2147483647"
LOCKEDPAGES="32:32"
SHMPAGES="8192:8192"
PRIVVMPAGES="22528:24576"
NUMFILE="2048:2048"
NUMFLOCK="100:110"
NUMPTY="16:16"
NUMSIGINFO="256:256"
DCACHESIZE="1048576:1097728"
PHYSPAGES="0:2147483647"
NUMIPTENT="128:128"
DISKSPACE="1048576:1153434"
DISKINODES="200000:220000"
QUOTATIME="0"
CPUUNITS="1000"
OFFLINE_MANAGEMENT="yes"
ARCH="x86"
PLATFORM="linux"
SLMMEMORYLIMIT="33521664:33521664"
```

get_virtual_config

Summary:

Converts virtual server configuration data from a virtualization technology native format to the Agent format.

Request specification:

Name	Min/Max	Type	Description
`get_virtual_config`			
`{`			
`native_config`	`1..1`	`native_configType (p. 47)`	Virtuozzo Container configuration data in the Virtuozzo native format.
`}`			

Returns:

Name	Min/Max	Type	Description
`virtual_config`	`0..[]`	`venv_configType` (p. 61)	Virtuozzo Container configuration data in the Agent format.

Description:

This call is an opposite of the `get_native_config` call (p. 228). It converts the Virtuozzo Container configuration data from the Virtuozzo native format to the Agent format.

event_log

Purpose:

Provides calls that allow to access event logs.

Types

audit_event_dataType

Summary:

Type specification:

Name	Min/Max	Type	Description
`session_id`	`0..1`	`string`	
`ip`	`0..1`	`ip_type (p. 26)`	
`user_name`	`1..1`	`string`	

Calls

Call	Description
get_events (p. 231)	Retrieves information from the event log.

get_events

Summary:

Retrieves information from the event log.

Request specification:

Name	Min/Max	Type	Description
`event_log`			
`{`			
`[`	`0..1`		Denotes a choice between the `eid` and the `subject` elements.

eid	1..1	eid_type (p. 25)	The ID of the server that generated the event. This is usually the Hardware Node hosting Virtuozzo Containers. When the event triggers in one of the Containers, it is actually generated by the Hardware Node and its ID is recorded in the log.
subject	1..1	eid_type (p. 25)	The ID of the affected server. This is usually the Container that triggered the event. For example, the Container the status of which has changed. The ID of the affected Container is also recorded in the log together with the server (host) that generated the event.
]			
start_time	0..1	datetime_type	Start time of the log.
end_time	0..1	datetime_type	End time of the log.
records	0..1	int	Number of records to retrieve from the end of the log.
sid	0..1	sidType (p. 27)	Report only the events with the SID specified in this element (the user SID identifies the active user at the time the event was generated).
source	0..1	string	Report only the events with the source specified here (the source is the name of the plug-in or an Agent operator that generated the event).
category	0..1	string	Report only the events associated with the category specified in this element. See the Types section (p. 504) for the available event types and their corresponding categories.
data	0..1	none	If this element is present, the event data will also be retrieved (the data element of the returned event structure will be populated with the event type-specific data).
}			

Returns:

Name	Min/Max	Type	Description
`event`	`0..[]`	`eventType (p. 36)`	Event information.

Description

Every time something changes in a server that may affect its operations (e.g. status or configuration change), a system event of a corresponding type is triggered and the event information is recorded into a log. The `get_events` call allows to retrieve event data from the log. You may specify any of the available parameters to narrow the search down and to retrieve only the information that you require. For more information on system events, see the Events chapter (p. 504).

Example:

Getting the latest 2 records from the event log database.

Input

```
<packet version="4.0.0" id="555">
  <target>event_log</target>
  <data>
    <event_log>
      <get_events>
          <records>2</records>
          <data/>
      </get_events>
    </event_log>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>event_log</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:data>
  <ns2:event_log>
    <ns2:event xsi:type="ns3:eventType">
      <ns3:eid>89e27960-97b8-461f-902f-557b4b16784b</ns3:eid>
      <ns3:time>2007-01-16T20:11:57+0000</ns3:time>
      <ns3:source>VZAConfPeriodic</ns3:source>
      <ns3:category>env_config</ns3:category>
      <ns3:sid>S-1-1000-3</ns3:sid>
      <ns3:count>1</ns3:count>
      <ns3:id>4b9ba735-6084-774c-9632-3f92bd905066</ns3:id>
      <ns3:info xsi:type="ns3:infoType">

<ns3:message>RW52aXJvbm1lbnQgJWVudiUgY29uZmlnIGNoYW5nZWQ=</ns3:message>
        <ns3:name></ns3:name>
        <ns3:translate/>
        <ns3:parameter>
          <ns3:message>JXRpdGxlJQ==</ns3:message>
          <ns3:name>env</ns3:name>
          <ns3:translate/>
          <ns3:parameter>

<ns3:message>NzIxNDViZjAtNzU2Mi00M2Q0LWI3MDctY2MzM2QzN2UzZjEw</ns3:message>
            <ns3:name>eid</ns3:name>
            <ns3:translate/>
          </ns3:parameter>
          <ns3:parameter>
            <ns3:message>bG9jYWxob3N0</ns3:message>
            <ns3:name>title</ns3:name>
            <ns3:translate/>
          </ns3:parameter>
          <ns3:parameter>
            <ns3:message>dmlydHVvenpv</ns3:message>
```

```
            <ns3:name>type</ns3:name>
            <ns3:translate/>
          </ns3:parameter>
        </ns3:parameter>
        <ns3:parameter>
          <ns3:message>JXRpdGxlJQ==</ns3:message>
          <ns3:name>source_env</ns3:name>
          <ns3:translate/>
          <ns3:parameter>

<ns3:message>ODllMjc5NjAtOTdiOC00NjFmLTkwMmYtNTU3YjRiMTY3ODRi</ns3:message>
            <ns3:name>eid</ns3:name>
            <ns3:translate/>
          </ns3:parameter>
          <ns3:parameter>
            <ns3:message>ZGhjcDAtMTk0LnN3LnJ1</ns3:message>
            <ns3:name>title</ns3:name>
            <ns3:translate/>
          </ns3:parameter>
          <ns3:parameter>
            <ns3:message>Z2VuZXJpYw==</ns3:message>
            <ns3:name>type</ns3:name>
            <ns3:translate/>
          </ns3:parameter>
        </ns3:parameter>
      </ns3:info>
    </ns2:event>
  </ns2:event_log>
</ns1:data>
</ns1:packet>
```

filer

Purpose:

The file management interface.

Types

credType

Summary:

User and group information.

Type specification:

Name	Min/Max	Type	Description
{			
[			
user	0..1	string	User name.
uid	0..1	long	User ID.
]			
[			

group	0..[]	string	Group name.
gid	0..[]	long	Group ID.
]			
umask	0..1	int	Operation umask.
}			

Description:

The <umask> parameter is used to restrict file system entries permission mode using the following rule:

[permission] AND ~[umask] => [result permission]

umask must specified using a decimal value, not octal.

navigateType

Summary:

Filesystem navigation.

Type specification:

Name	Min/Max	Type	Description
path	1..[]	base64Binary	Path on filesystem. If path is empty then look up partitions.
cwd	0..1	base64Binary	Current working directory. The default value is: "/" "C:\"
cred	0..1	credType (p. 235)	Credentials with which the requested operation will be performed.

navigate_wildType

Summary:

Filesystem navigation with wildcards.

Type specification:

Extends `navigateType (p. 236)`

Adds the following elements:

Name	Min/Max	Type	Description
`wildcard`	`0..1`	`none`	If present, indicates that the value of the `path` element (derived from the supertype `navigateType`) contains wildcards.

fileType

Summary:

Filesystem element.

Type specification:

Name	Min/Max	Type	Description
`file`	`0..[]`	`fileType (p. 237)`	If filesystem element is a directory, this field will contain the direct children of the directory (files and directories).
`name`	`1..1`	`base64Binary`	The filesystem element name.

type	0..1	int	Element type: 1 FIFO 2 Character device 4 Directory 6 Block device 8 Regular File 10 Symbolic link 12 Socket 16 Floppy disk 17 Hard disk 18 Remote drive 19 CD ROM 20 RAM disk 21 Mounted image file.
mode	0..1	mode_type (p. 239)	The element access mode in decimal form.
user	0..1	string	User name.
group	0..1	string	Group name.
uid	0..1	int	User ID.
gid	0..1	int	Group ID.
size	0..1	long	Element size.
date	0..1	datetime_type	The date when the file element was last changed.
links	0..1	int	Number of hard links.
link_name	0..1	string	Where link points to.
offset	0..[]	long	Offset in the file where the block of data was found by the search call (p. 261).
body	0..1	base64Binary	A buffer containing the file data.
content_type	0..1	string	File contents description (in MIME format).

mode_type

Summary:

File element access mode.

Type specification:

Simple type.

```
union: int, string
```

Calls

Call	Description
list (p. 239)	List directory contents.
remove (p. 244)	Removes filesystem entries.
copy (p. 245)	Copies files.
mkdir (p. 247)	Creates new directories.
move (p. 248)	Moves or renames files.
upload (p. 249)	Upload files into the specified server.
download (p. 251)	Downloads files from the specified server.
chmod (p. 254)	Change access permissions for all elements in the list.
chown (p. 255)	Change file owner and group.
link (p. 256)	Create new link element.
stat (p. 257)	Displays file or filesystem status.
readlink (p. 259)	Display value of a symbolic link.
search (p. 261)	Search the files for a block of data.

list

Summary:

Lists information about files, directories and other filesystem elements. The command is also capable of searching the backup archives and retrieve the information about the archived files and directories.

Request specification:

Name	Min/Max	Type	Description
`list`		`navigateType (p. 236)`	

{			
info	0..1		The fields for which to provide the output. If an element from this sequence is included in the call, the information that it refers to will be included in the result set.
{			
user	0..1		User name.
group	0..1		Group name.
uid	0..1		User ID.
gid	0..1		Group ID.
mode	0..1		Element mode.
size	0..1		Element size.
date	0..1		Date of last change.
links	0..1		Number of hard links.
link_name	0..1		Where the link points to.
content_type	0..1		File contents description.
}			
usage	0..1		If present, the size returned for an element is a gross size on a disk, so for directories it is calculated by traversing their children.
{			
single_fs	0..1		If included, the operation will not go across different partitions while traversing.
}			
follow_links	0..1		If present, the information returned for links will be about their references instead of themselves.
filter	0..[]		File filtering criteria. Inside a single filter, the AND rule applies (all must be satisfied). Multiple filters work as the OR rule (at least one should be satisfied).
{			

[			Denotes a choice between `user`, `uid`, or the `start_uid/end_uid` sequence.
user	1..1	string	User name (supports wildcards).
uid	1..1	int	User ID.
{			This sequence is not a child of the `uid` element but is a separate entry in the choice group.
start_uid	0..1	int	First UID of the range.
end_uid	0..1	int	Last UID of the range.
}			
]			
[			Denotes a choice between the `gid`, `group`, or the `start_uid/end_uid` sequence.
gid	1..1	int	Group ID.
group	1..1	string	Group name (supports wildcards).
{			*The sequence is not a child of the* `group` *element but a separate choice.*
start_uid	0..1	int	First UID of the range.
end_uid	0..1	int	Last UID of the range.
}			
]			
start_date	0..1	date	Time of the last change to start with.
end_date	0..1	date	Time of the last change to end with.
min_size	0..1	long	Minimum size.
max_size	0..1	long	Maximum size.
type	0..1	int	The filesystem element type. See `fileType/type` (p. 237) for the available types.
name	0..1	base64Binary	The element name (supports wildcards).

block	0..1	base64Binary	List files containing this text (this could be a time consuming operation).
}			
recursively	0..1	none	To list the entire tree including subdirectories include this element in the request.
}			

Returns:

Name	Min/Max	Type	Description
file	0..[]	fileType (p. 235)	The file information structure.

Description:

The path can be specified using wildcard extensions. Please note that if you are using a wildcard in any of the `path` elements, you must include the `wildcard` option. If an absolute path is given, the `cwd` parameter is ignored.

To search the backup archives, the `path` option must contain the URI specifying the location of the archived file or directory. The format of the URI is as follows:

```
backup://BACKUP_ID/path
```

where `backup` indicates that we want to search the backup archive; `BACKUP_ID` is a string containing the backup ID; `path` is the absolute path to the original location of a file or directory. For example:

```
<path>backup://2005-09-04T203847+0400@tc9/C:/Windows/info.txt</path>
```

If the `usage` option is included, the size returned for an element is the actual size on the disk (the size of the elements in blocks multiplied by the filesystem block size). For directories it is calculated by adding up the sizes of all the descendents of a directory.

You can customize the result set by specifying only the file properties that you want to see. This can be done by including the appropriate parameters in the `info` option.

The values of the `start_date` and the `end_date` elements are specified as a time in seconds starting from the year 1970. If `start_date` is absent, filter everything from 0 to the value specified in the `end_date` element. If `end_date` is absent, filter everything from `start_date` up to the current date.

Note: By default, the call will get the list of files from the Hardware Node. To retrieve the list from a Virtuozzo Container, use the remote message targeting mechanism by including the `dst` element in the message header containing the target Server ID. This rule applies to most of the file management calls.

 Example:

Retrieving a list of files from the "/" directory from the specified server.

Input

```
<packet version="4.0.0">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <list>
        <cwd>Lw==</cwd>
        <path>Lw==</path>
        <info>
          <user/>
          <name/>
          <uid/>
          <group/>
          <gid/>
        </info>
      </list>
    </filer>
```

```
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>filer</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:data>
  <ns2:filer>
    <ns2:file>
      <ns2:name>aG9tZQ==</ns2:name>
      <ns2:user>root</ns2:user>
      <ns2:group>root</ns2:group>
      <ns2:uid>0</ns2:uid>
      <ns2:gid>0</ns2:gid>
    </ns2:file>
    <ns2:file>
      <ns2:name>bGli</ns2:name>
      <ns2:user>root</ns2:user>
      <ns2:group>root</ns2:group>
      <ns2:uid>0</ns2:uid>
      <ns2:gid>0</ns2:gid>
    </ns2:file>
    <ns2:file>
      <ns2:name>cHJvYw==</ns2:name>
      <ns2:user>root</ns2:user>
      <ns2:group>root</ns2:group>
      <ns2:uid>0</ns2:uid>
      <ns2:gid>0</ns2:gid>
    </ns2:file>
    <ns2:file>
      <ns2:name>Li4=</ns2:name>
      <ns2:user>root</ns2:user>
      <ns2:group>root</ns2:group>
      <ns2:uid>0</ns2:uid>
      <ns2:gid>0</ns2:gid>
    </ns2:file>
    <ns2:file>
      <ns2:name>bWlzYw==</ns2:name>
      <ns2:user>root</ns2:user>
      <ns2:group>root</ns2:group>
      <ns2:uid>0</ns2:uid>
      <ns2:gid>0</ns2:gid>
    </ns2:file>
    <ns2:file>

    <!-- the rest of the output is omitted -->
  </ns2:filer>
</ns1:data>
</ns1:packet>
```

remove

Summary:

Removes filesystem entries.

Request specification:

Name	Min/Max	Type	Description
remove	1..1	navigateType (p. 236)	
{			

recursively	0..1	none	Include this element in the request if you are removing a directory.
force	0..1	none	Ignore errors if a file or a directory does not exist.
}			

Returns:

OK/Error

Example:

Removing the specified directory from the specified server. When performing the operation on a Virtuozzo Container, specify it's Server ID using the remote message targeting mechanism (the `dst` element in the message header).

Input

```
<packet>
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <remove>
        <cwd>Lw==</cwd>
        <path>dGVzdGRpcg==</path>
         <recursively/>
      </remove>
    </filer>
  </data>
</packet>
```

copy

Summary:

Copies files and directories.

Request specification:

Name	Min/Max	Type	Description
copy		navigateType (p. 236)	The parent navigate_wildType type contains parameters allowing to specify source file information.
{			
dst_path	1..1	base64Binary	Destination path including the directory name and the file name.

recursively	0..1		A flag indicating to process directories recursively.
force	0..1		A flag indicating to overwrite the file if it exists in the target directory.
{	0..1		Remote copying. *The sequence is not a child of the* `force` *element but a separate sequence.*
dst	1..1	connection_infoType (p. 30)	The destination Container connection information. The Container must be located on the same host.
mode	0..1	mode_type (p. 239)	New access mode (in decimal form) to assign to copied files on the target machine.
}			
}			

Returns:

OK/Error

Description:

You can copy files between directories on the same server. You can also copy files between Virtuozzo Containers on the same host. To copy a file from one Container to another, include the `dst` element specifying the destination Server ID and, if necessary, the `mode` element specifying the new file access mode.

Example:

Copies a file `mylog.log` from the `/root` directory to the `/root/mylogs` directory using `root/root` credentials.

```
<packet>
  <target>filer</target>
  <data>
    <filer>
      <copy>
        <path>cm9vdC9teWxvZy5sb2c=</path>
        <cwd>Lw==</cwd>
        <cred>
          <user>root</user>
          <group>root</group>
        </cred>
        <dst_path>L3Jvb3QvbXlsb2dzL215bG9nLmxvZw==</dst_path>
      </copy>
    </filer>
  </data>
</packet>
```

mkdir

Summary:

Creates a new directory.

Request specification:

Name	Min/Max	Type	Description
`mkdir`		`navigateType (p. 236)`	
`{`			
`mode`	`0..1`	`mode_type (p. 239)`	Access mode in decimal form.
`recursively`	`0..1`		If any part of the path does not exist, create it.
`}`			

Returns:

OK/Error

Example:

```
<packet>
  <target>filer</target>
  <data>
    <filer>
      <mkdir>
        <path>cm9vdC9teWxvZ3M=</path>
        <cwd>Lw==</cwd>
        <cred>
          <user>root</user>
          <group>root</group>
        </cred>
      </mkdir>
    </filer>
  </data>
</packet>
```

move

Summary:

Moves or renames files.

Request specification:

Name	Min/Max	Type	Description
move		navigateType (p. 236)	
{			
dst_path	1..1	base64Binary	The destination path.
force	0..1		Overwrite if destination exists.
}			

Returns:

OK/Error

Description:

To rename a file or a directory, specify the new name in the `dst_path`. To copy files to a different directory, specify the full directory path.

upload

Summary:

Uploads a file to a server.

Request specification:

Name	Min/Max	Type	Description
`upload`			
`{`			

`cwd`	`0..1`	`base64Binary`	Current working directory. The default value is:
			 `"/"`
			 `"C:\"`

`cred`	`0..1`	`credType (p. 235)`	Credentials with which the requested operation will be performed.
`file`	`1..[]`		Files to upload. One packet may contain multiple `file` elements so you can transfer more than one block of data in one call.
`{`			
`path`	`1..1`	`base64Binary`	Target path and file name.
`size`	`1..1`	`long`	The size of the data block being transferred in bytes.
`offset`	`0..1`	`long`	Offset in the target file in bytes. If not supplied, then the data will be inserted at the beginning of the file. If set to `-1`, then the data will be appended at the end of the file. The `force` element (below) must be always included when working with existing files.

body	0..1	base64Binary	The block of data to be transferred.
}			
mode	0..1	mode_type (p. 239)	Access permissions for the new file in decimal form. The mode is affected by umask.
force	0..1	none	Include this element if the destination file already exists and you want to overwrite it or add more data to it using the offset option. If the element is absent and the file already exists on the destination machine, the call will fail.
}			

Returns:

OK/Error

Example:

Input

Uploading the first block of data. The file doesn't exist on the target machine yet, so it will be created.

```
<packet version="4.0.0" id="545">
  <target>filer</target>
  <data>
    <filer>
      <upload>
        <file>
          <path>dGVzdDAxLnR4dA==</path>
          <size>12</size>
          <body>RGF0YSBibG9jayAx</body>
        </file>
      </upload>
    </filer>
  </data>
</packet>
```

Uploading the second data block. To append the data to the end of the file, the `offset` and the `force` options must be used.

```
<packet version="4.0.0" id="545">
  <target>filer</target>
  <data>
    <filer>
      <upload>
        <file>
          <path>dGVzdDAxLnR4dA==</path>
          <size>17</size>
          <body>U2Vjb25kIGRhdGEgYmxvY2s=</body>
          <offset>12</offset>
        </file>
        <force/>
      </upload>
    </filer>
  </data>
</packet>
```

download

Summary:

Downloads a file from a server. The call is also capable of extracting files from a backup archive.

Request specification:

Name	Min/Max	Type	Description
`download`			
{			

cwd	0..1	base64Binary	Current working directory. The default value is: "/" "C:\"
cred	0..1	credType (p. 235)	Credentials with which the requested operation will be performed.
file	1..[]		Files to download.
{			
path	1..1	base64Binary	The source path and file name.
size	0..1	long	Size of the data block to be transferred.
offset	0..1	long	Offset in the source file.
}			
}			

Returns:

Name	Min/Max	Type	Description
file			
{			
body	0..1	base64Binary	The file data.
}			

Description:

The call reads the data from the specified source file and transfers the data to the client in blocks of the specified size. It does not automatically create a file on the target machine, so it is your responsibility to process the received data.

To extract a file from a backup archive, the <cwd> element must contain a URI specifying the backup ID. The format of the URI is as follows:

```
backup://backup_id
```

where backup_id is the ID of the backup archive, and the <path> element must contain an absolute path to the original location of a file or directory. For example:

```
<cwd>backup://a28d77df-a4e1-4d98-a01c-dc85b6d19f7b/20060718064512</cwd>
<path>C:/Windows/info.txt</path>
```

Example:

Downloading a file from a server in two sequential transfers.

Input

Reading the first block of data.

```
<packet version="4.0.0" id="2">
  <target>filer</target>
  <data>
    <filer>
      <download>
        <file>
          <path>dGVzdDAxLnR4dA==</path>
          <size>12</size>
        </file>
      </download>
    </filer>
  </data>
</packet>
```

Output

The body element contains the data.

```
<ns1:packet version="4.0.0">
<ns1:origin>filer</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:filer>
    <ns2:file>
      <ns2:body>RGF0YSBibG9jayAx</ns2:body>
    </ns2:file>
  </ns2:filer>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>

Input
```

Reading the second block of data. The `offset` element marks the beginning of the block in the source file.

```
<packet version="4.0.0" id="2">
  <target>filer</target>
  <data>
    <filer>
      <download>
        <file>
          <path>dGVzdDAxLnR4dA==</path>
          <size>17</size>
          <offset>12</offset>
        </file>
      </download>
    </filer>
  </data>
</packet>
```

Output

The body element contains the data.

```
<ns1:packet version="4.0.0">
  <ns1:origin>filer</ns1:origin>
  <ns1:target>vzclient3</ns1:target>
  <ns1:data>
    <ns2:filer>
      <ns2:file>
        <ns2:body>U2Vjb25kIGRhdGEgYmxvY2s=</ns2:body>
      </ns2:file>
    </ns2:filer>
  </ns1:data>
</ns1:packet>
```

chmod

Summary:

Change access permissions for filesystem elements.

Request specification:

Name	Min/Max	Type	Description
chmod		navigateType (p. 236)	
{			
mode	1..1	mode_type (p. 239)	New access mode in decimal form.
[	0..1		Turn on/off bits supplied in mode. Analogues to +/- in chmod command.
on	1..1	none	
off	1..1	none	
]			
recursively	0..1	none	Apply changes recursively.
}			

Returns:

OK/Error

chown

Summary:

Changes the user and/or group ownership of a specified file.

Request specification:

Name	Min/Max	Type	Description
`chown`		`navigateType (p. 236)`	
`{`			
`recursively`	`0..1`		Process elements recursively.
`owner`	`1..1`	`credType (p. 235)`	Owner name (as it exists in `/etc/passwd`).
`follow_links`	`0..1`		Operates as `lchown`.
`}`			

Returns:

OK/Error

Description:

New owner/group information is supplied as a set of `user` or `uid` and/or `group` or `gid` within the `owner` structure.

The `user` element can be used to specify a user or a group.

Example:

Changing the owner of /test to apache/apache.

```
<packet version="4.0.0" id="324">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <chown>
        <path>dGVzdA==</path>
        <cwd>Lw==</cwd>
        <cred>
          <user>apache</user>
          <group>apache</group>
        </cred>
      </chown>
    </filer>
  </data>
</packet>
```

link

Summary:

Creates a link to the specified filesystem entry.

Request specification:

Name	Min/Max	Type	Description
`link`		`navigateType (p. 236)`	
`{`			
`name`	`1..1`	`base64Binary`	Link name.
`symbolic`	`0..1`		Creates symbolic link if present, otherwise creates hard link.
`force`	`0..1`		Ignores errors if present.
`}`			

Returns:

OK/Error

Example:

Creating a symbolic link to `/test`.

```
<packet version="4.0.0" id="356">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <link>
        <cwd>Lw==</cwd>
        <path>dGVzdA==</path>
        <name>dGVzdC1kaXI=</name>
        <symbolic/>
        <force/>
      </link>
    </filer>
  </data>
</packet>
```

stat

Summary:

Returns filesystem element status information. The call is also capable of searching the backup archives and display the status information for the archived files and directories.

Request specification:

Name	Min/Max	Type	Description
stat		navigateType (p. 236)	
{			
usage	0..1		If present, the size returned for an element is a gross size on a disk, so for directories it is calculated by traversing their children.
{			
single_fs	0..1		Do not go across different partitions, while traversing.
}			
follow_links	0..1		If present, the information returned for links will be about their references, instead of themselves.

info	0..1		The list of properties to retrieve the information for. Simply include any of the following elements in the request if you would like the corresponding information to be included in the output.
{			
user	0..1	none	User name.
group	0..1	none	Group name.
uid	0..1	none	User ID.
gid	0..1	none	Group ID.
mode	0..1	none	Mode.
size	0..1	none	Size.
date	0..1	none	Date.
links	0..1	none	Hard link information.
link_name	0..1	none	Link name.
content_type	0..1	none	File element content type.
}			

Returns:

Name	Min/Max	Type	Description
file	0..[]	fileType (p. 237)	Status information.

Example:

Get the status of the "/" directory.

Input

```
<packet version="4.0.0" progress="on">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <stat>
        <cwd>Lw==</cwd>
        <path>Lw==</path>
      </stat>
    </filer>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>filer</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:data>
  <ns2:filer>
    <ns2:file>
      <ns2:name>Lw==</ns2:name>
      <ns2:type>4</ns2:type>
      <ns2:mode>493</ns2:mode>
      <ns2:user>root</ns2:user>
      <ns2:group>root</ns2:group>
      <ns2:uid>0</ns2:uid>
      <ns2:gid>0</ns2:gid>
      <ns2:links>27</ns2:links>
      <ns2:size>0</ns2:size>
      <ns2:date>2007-01-24T01:50:25+0000</ns2:date>
    </ns2:file>
  </ns2:filer>
</ns1:data>
</ns1:packet>
```

readlink

Summary:

Returns the contents of a symbolic link.

Request specification:

Name	Min/Max	Type	Description
`readlink`		`credType` (p. 235)	
{			
`path`	`1..[]`	`base64Binary`	Path on filesystem.
`cwd`	`0..1`	`base64Binary`	Current working directory. If absent, it is set to "/" by default.

`}`			

Returns:

Name	Min/Max	Type	Description
`file`	`0..[]`	`fileType (p. 237)`	File information structure. If the path is a symbolic link, returns its contents (what it links to) in `file/name`. Returns an error otherwise. On systems that do not support symbolic links, the operation will always return an error.

Description:

If the path is a symbolic link, returns its contents (what it links to) in `file/name` Returns an error otherwise. On systems that do not support symbolic links, the operation will always return an error.

Example:

Input

```
<packet version="4.0.0">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <readlink>
        <veid>103</veid>
        <path>L2hvbWUvZWR2dC9kLWVkdnQ=</path>
      </readlink>
    </filer>
  </data>
</packet>
```

Output

```
<packet priority="0" version="4.0.0">
  <origin>filer</origin>
  <target>vzclient84</target>
  <data>
    <filer>
      <file>
        <name>L2hvbWUvZWR2dA==</name>
      </file>
    </filer>
  </data>
</packet>
```

search

Summary:

Finds the first occurrence of a data block in files.

Request specification:

Name	Min/Max	Type	Description
`search`		`navigateType (p. 236)`	
`{`			
`offset`	`0..1`	`long`	Starting search offset. Positive offset means offset from the beginning of a file, and negative from the end of a file.
`block`	`1..1`	`base64Binary`	Data block to find in a file.
`backward`	`0..1`		Search backwards.

}			

Returns:

Name	Min/Max	Type	Description
file	0..[]		
{			
name		fileType (p. 237)	The name of the file containing the specified block of data.
offset	0..[]	long	Offset at which the block is located in the file.
}			

Example:

Search for a string "aaa" in the `/tmp.txt` file on the specified Virtuozzo Container.

Input

```
<packet version="4.0.0">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <search>
        <cwd>Lw==</cwd>
        <path>L3RtcC50eHQ=</path>
        <offset>0</offset>
        <block>YWFhYQ==</block>
      </search>
    </filer>
  </data>
</packet>
```

Output

```
<packet version="4.0.0">
  <origin>filer</origin>
  <target>vzclient123</target>
  <data>
    <filer>
      <file>
        <name>dG1wLnR4dA==</name>
        <offset>15</offset>
      </file>
    </filer>
   </data>
</packet>
```

firewallm

This interface is available on Linux-based systems only.

Purpose:

The firewall management interface.

Types

chainType

Summary:

An enumeration defining the firewall chain types.

Type specification:

Enumeration, `string`.

Enumerator	Description

INPUT	An input chain. Called for inbound packets only.
OUTPUT	An output chain. Called for outbound packets only.
FORWARD	Forward chain. Called for packets that pass through the system.

policyType

Summary:

An enumeration defining the firewall control policy types.

Type specification:

Enumeration, `string`.

Enumerator	Description
ACCEPT	Accept a packet.
DROP	Drop a packet.
REJECT	Reject a packet.

port_rangeType

Summary:

This type is used to specify a single TCP port or a range of TCP ports.

Type specification:

Name	Min/Max	Type	Description
first_port	1..1	int	The first port number in the range. If specifying a single port, include it here and omit the last_port element (below).
last_port	0..1	int	The last port number in the range.

ruleType

Summary:

Contains the firewall rule settings.

Type specification:

Name	Min/Max	Type	Description
name	1..1	string	Name of the service.
protocol	1..1	transport_type (p. 27)	Protocol type (tcp or udp).
chain	0..1	chainType (p. 263)	The firewall chain to which this rule is applied.
policy	0..1	policyType (p. 264)	Firewall security policy.
allowed	0..1	boolean	Indicates whether the protocol is enabled or disabled when the firewall is switched on. true -- the protocol will be enabled. false -- the protocol will be disabled.
src_addr	0..1	net_addressType	Source address. The internal address of the packets (e.g.: IPv4 or IPv6 address, the name of a network interface, etc.)
dst_addr	0..1	net_addressType	Destination address. The address where the packets are sent.
src_ports	0..1	port_rangeType (p. 264)	Source port range.
dst_ports	0..1	port_rangeType (p. 264)	Destination port range.
input_iface	0..1	string	Input interface.
output_iface	0..1	string	Output interface.

Calls

Call	Description
get (p. 266)	Lists existing firewall rules.
set (p. 267)	Adds or modifies a rule.
delete (p. 268)	Deletes a rule.

is_enabled (p. 269)	Checks if the firewall is enabled.
enable (p. 270)	Enables the firewall.
disable (p. 271)	Disables the firewall.

get

Summary:

Lists existing firewall rules.

Request specification:

Name	Min/Max	Type	Description
`get`	`0..1`		

Returns:

Name	Min/Max	Type	Description
`rule`	`1..[]`	`ruleType (p. 265)`	Firewall rule structure.

Example:

Listing firewall rules for the specified Virtuozzo Container.

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>firewallm</target>
  <data>
    <firewallm>
      <get/>
    </firewallm>
  </data>
</packet>
```

Output

```
<packet priority="0" version="4.0.0">
  <origin>firewallm</origin>
  <target>vzclient30</target>
  <data>
    <firewallm>
      <rule>
        <name>REJECT_MAIL</name>
        <chain>INPUT</chain>
        <policy>REJECT</policy>
        <allowed>1</allowed>
        <protocol>tcp</protocol>
        <src_addr>
          Host192.168.1.10</host>
          <mask>255.255.255.0</mask>
        </src_addr>
        <src_ports>
          <first_port>25</first_port>
        </src_ports>
      </rule>
    </firewallm>
  </data>
</packet>
```

set

Summary:

Adds a rule to the firewall, or bans/permits an existing rule.

Request specification:

Name	Min/Max	Type	Description
`set`			
`{`			
`rule`	`0..[]`	`ruleType` (p. 265)	Firewall rule to set.
`}`			

Returns:

OK/Error

Example:

Setting a new rule for the specified Virtuozzo Container.

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>firewallm</target>
  <data>
    <firewallm>
      <set>
        <rule>
          <name>REJECT_MAIL</name>
          <protocol>tcp</protocol>
          <chain>INPUT</chain>
          <policy>REJECT</policy>
          <allowed>1</allowed>
          <src_addr>
            Host192.168.2.10</host>
            <mask>255.255.255.0</mask>
          </src_addr>
          <src_ports>
            <first_port>25</first_port>
          </src_ports>
        </rule>
      </set>
    </firewallm>
  </data>
</packet>
```

delete

Summary:

Deletes a firewall rule.

Request specification:

Name	Min/Max	Type	Description
`delete`			
`{`			
`rule`	`0..[]`	`ruleType` (p. 265)	Firewall rule to delete. If none specified, deletes all existing rules.
`}`			

Returns:

OK/Error

Example:

Deleting a firewall rule from the specified Container. All of the parameters shown in the example must be specified.

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>firewallm</target>
  <data>
    <firewallm>
      <delete>
        <rule>
          <name>REJECT_MAIL1</name>
          <chain>INPUT</chain>
          <policy>REJECT</policy>
          <allowed>1</allowed>
          <protocol>tcp</protocol>
          <src_addr>
            Host192.168.2.10</host>
            <mask>255.255.255.0</mask>
          </src_addr>
          <src_ports>
            <first_port>25</first_port>
          </src_ports>
        </rule>
      </delete>
    </firewallm>
  </data>
</packet>
```

is_enabled

Summary:

Checks if the firewall is enabled.

Request specification:

Name	Min/Max	Type	Description
`is_enabled`	`0..1`		

Returns:

Name	Min/Max	Type	Description
`status`	`0..1`	`boolean`	`true` -- the firewall is enabled. `false` -- the firewall is disabled.

Example:

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>firewallm</target>
  <data>
    <firewallm>
      <is_enabled/>
    </firewallm>
  </data>
</packet>
```

Output:

```
<ns1:packet priority="0" version="4.0.0">
  <ns1:origin>firewallm</ns1:origin>
  <ns1:target>vzclient3</ns1:target>
  <ns1:data>
    <ns2:firewallm>
      <ns2:status>0</ns2:status> <!-- disabled -->
    </ns2:firewallm>
  </ns1:data>
</ns1:packet>
```

enable

Summary:

Enables a firewall.

Request specification:

Name	Min/Max	Type	Description
enable	0..1		

Returns:

OK/Error

Example:

Enabling a firewall in the specified Container.

```
<packet version="4.0.0">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>firewallm</target>
  <data>
    <firewallm>
      <enable/>
    </firewallm>
  </data>
</packet>
```

disable

Summary:

Disables a firewall.

Request specification:

Name	Min/Max	Type	Description
disable	0..1		

Returns:

OK/Error

Example:

Disabling a firewall in the specified Container.

```
<packet version="4.0.0">
  <dst>
    Host24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>firewallm</target>
  <data>
    <firewallm>
      <disable/>
    </firewallm>
  </data>
</packet>
```

licensem

Purpose:

Virtuozzo Containers license management interface.

Types

parameterType

Summary:

Contains information about a license parameter.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	Parameter name.
`value`	`1..1`	`base64Binary`	Parameter value.
`used`	`0..1`	`base64Binary`	Number of licensable units already in use. For example, the number of Virtuozzo Containers in use. By subtracting `used` from `value` you'll get the number of Containers that you still can create under the given license. The value is only provided for some of the license parameters.

licenseType

Summary:

Contains a list of license parameters.

Type specification:

Name	Min/Max	Type	Description
parameter	0..[]	parameterType (p. 272)	A list of license parameters.

licensesType

Summary:

License information.

Type specification:

Name	Min/Max	Type	Description
body	1..1	string	License body.
license	1..[]	licenseType (p. 273)	License information.

license_eventType

Summary:

Event-specific data returned in the event of license expiration. See subscribe (p. 558) for more information on events and subscriptions.

Type specification:

Name	Min/Max	Type	Description
license	1..[]		
{			
serial	1..1	base64Binary	License serial number.
status	0..1	base64Binary	License status. If absent then the license was deleted.
}			

Calls

Call	Description
list (p. 274)	Retrieves a list of installed licenses.
install (p. 277)	Installs a new license.
parse (p. 281)	Parse license provided and get it's info.
delete (p. 284)	Delete license from the host server.
get_hwid (p. 285)	Gets Hardware Node HWID(s).
update	Updates current license.

list

Summary:

Retrieves the Virtuozzo Containers license installed on the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
`list`	`1..1`		
`{`			
`  serial`	`0..[]`	`string`	This parameter is not currently used.
`  type`	`0..[]`	`string`	This parameter is not currently used.
`}`			

Returns:

Name	Min/Max	Type	Description
`licenses`	`0..[]`	`licensesType (p. 273)`	Virtuozzo Containers license information.

Example:

Input

```
<packet version="4.0.0">
  <target>licensem</target>
  <data>
    <licensem>
      <list/>
    </licensem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/licensem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="1c4794742dt4823r9b8" time="2008-01-21T14:06:59+0000">
<origin>licensem</origin>
<target>vzclient2-9634f0ee-8c54-924d-9193-64f79dfc8736</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <licensem>
    <licenses>

<body>================================================================================
== This file contains license for Virtuozzo product.                          ==
== Please visit http://www.sw-soft.com for more information on                ==
== Virtuozzo and SWsoft.                                                      ==
== Please email your questions and/or suggestions to                          ==
== sales@sw-soft.com                                                          ==
============================ Start of license ============================
+hLwC8cTKogM7uox6MfKIwYGgk/eYX5KAxiTGNN0PsmvwSkG3L+UVC7oT6VoN0xQlAd7GhIk
6JuvxGvTpyDKbA22JfBiftDv1hK0X9p6OoaMdmvnEAL/peUfmcnbAR+JgBJu4+x8OeUNWftY
B6SlA8mt5ZONKQWzvT6il8rqdnDzgBevNQ+GcptCpPhibUtzP0x3ocKApGtKseAP1QK5zW4F
Y8LMds8dgrtjVe8tPUFXAnAzDkrpM0ugwqlimp2XwDA0ZXFHCdI1TN2L1Nehzng/xQ/a1Wmw
YyyHaV9irXtysOMT+D69Jrg1Jgcvz5i7uv8jIBWIaqhbu0Gqje2bTNjAEOWzdYrpeWZn2vYj
gGThn5QUlZArDt2Cg4UggTSLq4C1uQBylXlX/899v4AiT5zy9o5xfFlqzK+B9mofR8IBTI3+
B9QzTehiXqE4Ry2bsLgyxfbSWuF3cFx23Mdt/Um+fB9PF/OgqnsifAKSqjSApN1eQNFDERbm
oUEKAkIVCbtLcmHGpGETsWVMuWLGvQzkOidwJEIThYiO+Lon82qmMZkZ38TUUjC25pRr+Jia
+GdzO6Rr4bN8cqsZkb0Hw0SN3lRff+P/D56yzxNYlxeF6xfb99ynGQ4hDuoLQcgdLxIjX/aE
vkIliXHmVZVpQINIyxzfGoUIpNmc3X9GNcUl2vfRLtAyQm4h9vo8eQ0m8qn8wrEya3DOHcu8
duJ5/6ZyTrkiUhuhz1KQUco4cQiFe4aEn3Tjvvv1P/RHcJ8p7q0eAkOoIvFhyfWwKSkApfn2
+475QB0wD/rUci/bDj2TALdW0a0xRL0kFYq9JhJUy6rkvYUKjzLj0qzmVWdcQtpSwWjJrTds
22eJfG0Po6NL4+bQaC/qDJ1leK132DYeAWvgzjvP8iB7527swde67mvcfzALbJtXRaXpmlyL
qfzyzaVLkTEhgg6ffYDQpYSO7jgY1KF41Q+jPsqQp1/Kn7HfOlShPvDExfKlB6OHpQPu50om
oZXs27l7u6jwKhTbSd4bvDUXGyZWKHA=
============================ End of license ============================
      </body>
      <license>
        <parameter>
          <name>CLASS</name>
          <value>VlpTUlY=</value>
        </parameter>
        <parameter>
          <name>owner_name</name>
          <value>RXZhbHVhdGlvbiBWejQuMCBzZXJ2ZXIgSQ==</value>
        </parameter>
        <parameter>
          <name>status</name>
          <value>QUNUSVZF</value>
        </parameter>
        <parameter>
          <name>version</name>
```

```
          <value>NC4w</value>
        </parameter>
        <parameter>
          <name>owner_id</name>
          <value>MTEuMg==</value>
        </parameter>
        <parameter>
          <name>hwid</name>
          <value>MDAwMC4wMDAwLjAwMDAuMDAwMC4wMDAwLjAwMDAuMDAwMC4wMDAw</value>
        </parameter>
        <parameter>
          <name>serial</name>
          <value>NTc4RS43RjE3LkE5MEYuNzQyMS45RUU3LkRBOTIuMkVERC42ODI5</value>
        </parameter>
        <parameter>
          <name>expiration</name>
          <value>MjAwOC0wMi0wMVQwMDowMDowMCswMDAw</value>
        </parameter>
        <parameter>
          <name>start_date</name>
          <value>MjAwNi0wOS0wMVQwMDowMDowMCswMDAw</value>
        </parameter>
        <parameter>
          <name>issue_date</name>
          <value>MjAwNy0xMi0yN1QwNzowNjozMCswMDAw</value>
        </parameter>
        <parameter>
          <name>graceperiod</name>
          <value>MzAw</value>
          <used>MzAw</used>
        </parameter>
        <parameter>
          <name>key_number</name>
          <value>VlouMDAwMDAwMTMuMDAwNA==</value>
        </parameter>
        <parameter>
          <name>key_number_version</name>
          <value>MDAwNA==</value>
        </parameter>
        <parameter>
          <name>key_number_value</name>
          <value>MDAwMDAwMTM=</value>
        </parameter>
        <parameter>
          <name>product</name>
          <value>VmlydHVvenpv</value>
        </parameter>
        <parameter>
          <name>ct_total</name>
          <value>ODE5Mg==</value>
          <used>MA==</used>
        </parameter>
        <parameter>
          <name>cpu_total</name>
          <value>NjQ=</value>
          <used>MQ==</used>
        </parameter>
        <parameter>
          <name>vzpp_allowed</name>
          <value>MQ==</value>
        </parameter>
        <parameter>
          <name>backup_mgmt_allowed</name>
          <value>MQ==</value>
        </parameter>
        <parameter>
          <name>workflow_mgmt_allowed</name>
```

```
          <value>MQ==</value>
        </parameter>
        <parameter>
          <name>vzagent_allowed</name>
          <value>MQ==</value>
        </parameter>
        <parameter>
          <name>max_vzmc_users</name>
          <value>MTI4</value>
        </parameter>
        <parameter>
          <name>max_vzcc_users</name>
          <value>MjU2</value>
        </parameter>
        <parameter>
          <name>architecture</name>
          <value>eDg2</value>
        </parameter>
        <parameter>
          <name>architecture</name>
          <value>eDg2XzY0</value>
        </parameter>
        <parameter>
          <name>architecture</name>
          <value>aWE2NA==</value>
        </parameter>
        <parameter>
          <name>platform</name>
          <value>YW55</value>
        </parameter>
      </license>
    </licenses>
  </licensem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

install

Summary:

Installs a Virtuozzo Containers license on the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
install	1..1		
{			
license	1..1	string	License body, product key, or product activation code, whichever is available to you.
options	1..1		Installation options.
{			

transfer	0..[]	boolean	Include this parameter and set it to true if you would like to transfer the license to a different Hardware Node. The parameter can only be used when the license element contains a product activation code. If you have a license file or a product key, delete the license first using the delete call (p. 284) and then install it on another Hardware Node using the install call in a usual manner.
}			
}			

Returns:

Name	Min/Max	Type	Description
licenses	0..[]	licensesType (p. 273)	License information.

Example:

Installing a Virtuozzo Containers license using a product key.

Input

```
<packet version="4.0.0">
  <target>licensem</target>
  <data>
    <licensem>
      <install>
        <license>0M9Y96-2GVKXG-82773A-BBKHYB-VFDYDP</license>
      </install>
    </licensem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/licensem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="10c464846a5t5f90re80" time="2007-05-15T10:23:28+0000" priority="0"
version="4.0.0">
<ns1:origin>licensem</ns1:origin>
<ns1:target>vzclient2</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:licensem>
    <ns2:license>
      <ns2:parameter>
        <ns2:name>CLASS</ns2:name>
        <ns2:value>VlpTUlY=</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>status</ns2:name>
        <ns2:value>VkFMSUQ=</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>version</ns2:name>
        <ns2:value>NC4w</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>serial</ns2:name>

<ns2:value>ME05WTk2LTJHVktYRy04Mjc3M0EtQkJLSFlCLVZGRFlEUA==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>expiration</ns2:name>
        <ns2:value>MjAwNy0wNi0wMVQyMzo1OTo1OSswMDAw</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>graceperiod</ns2:name>
        <ns2:value>MzYwMA==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>key_number</ns2:name>
        <ns2:value>VlouMDAwMDAwMTMuMDAwMA==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>cpu_total</ns2:name>
        <ns2:value>NjQ=</ns2:value>
      </ns2:parameter>
```

```
      <ns2:parameter>
        <ns2:name>ve_total</ns2:name>
        <ns2:value>ODIwMA==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>max_vzmc_users</ns2:name>
        <ns2:value>MTI4</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>max_vzcc_users</ns2:name>
        <ns2:value>MjYw</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>platform</ns2:name>
        <ns2:value>QW55</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>product</ns2:name>
        <ns2:value>VmlydHVvenpv</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>vzpp_allowed</ns2:name>
        <ns2:value>MQ==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>backup_mgmt_allowed</ns2:name>
        <ns2:value>MQ==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>workflow_mgmt_allowed</ns2:name>
        <ns2:value>MQ==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>vzagent_allowed</ns2:name>
        <ns2:value>MQ==</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>architecture</ns2:name>
        <ns2:value>eDg2</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>architecture</ns2:name>
        <ns2:value>eDg2XzY0</ns2:value>
      </ns2:parameter>
      <ns2:parameter>
        <ns2:name>architecture</ns2:name>
        <ns2:value>aWE2NA==</ns2:value>
      </ns2:parameter>
    </ns2:license>
  </ns2:licensem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

parse

Summary:

Parses a license body and obtains the license parameters.

Request specification:

Name	Min/Max	Type	Description
parse	1..1		
{			
body	1..1	string	Base64-encoded license body.
type	0..[]	string	This parameter is not currently used.
serial	0..[]	string	This parameter is not currently used.
}			

Returns:

Name	Min/Max	Type	Description
licenses	0..[]	licensesType (p. 273)	License information.

Example:

Input

```
<packet version="4.0.0">
  <target>licensem</target>
  <data>
    <licensem>
      <parse>

<body>================================================================================
== This file contains license for Virtuozzo product.                          ==
== Please visit http://www.sw-soft.com for more information on                ==
== Virtuozzo and SWsoft.                                                      ==
== Please email your questions and/or suggestions to                          ==
== sales@sw-soft.com                                                          ==
============================ Start of license ============================
+hLwC8cTKogM7uox6MfKIwYGgk/eYX5KAxiTGNN0PsmvwSkG3L+UVC7oT6VoN0xQlAd7GhIk
6JuvxGvTpyDKbA22JfBiftDv1hK0X9p6OoaMdmvnEAL/peUfmcnbAR+JgBJu4+x8OeUNWftY
B6SlA8mt5ZONKQWzvT6il8rqdnDzgBevNQ+GcptCpPhibUtzP0x3ocKApGtKseAP1QK5zW4F
Y8LMds8dgrtjVe8tPUFXAnAzDkrpM0ugwqlimp2XwDA0ZXFHCdI1TN2L1Nehzng/xQ/a1Wmw
YyyHaV9irXtysOMT+D69Jrg1Jgcvz5i7uv8jIBWIaqhbu0Gqje2bTNjAEOWzdYrpeWZn2vYj
gGThn5QUlZArDt2Cg4UggTSLq4C1uQBylXlX/899v4AiT5zy9o5xfFlqzK+B9mofR8IBTI3+
B9QzTehiXqE4Ry2bsLgyxfbSWuF3cFx23Mdt/Um+fB9PF/OgqnsifAKSqjSApN1eQNFDERbm
oUEKAkIVCbtLcmHGpGETsWVMuWLGvQzkOidwJEIThYiO+Lon82qmMZkZ38TUUjC25pRr+Jia
+GdzO6Rr4bN8cqsZkb0Hw0SN3lRff+P/D56yzxNYlxeF6xfb99ynGQ4hDuoLQcgdLxIjX/aE
vkIliXHmVZVpQINIyxzfGoUIpNmc3X9GNcUl2vfRLtAyQm4h9vo8eQ0m8qn8wrEya3DOHcu8
duJ5/6ZyTrkiUhuhz1KQUco4cQiFe4aEn3Tjvvv1P/RHcJ8p7q0eAkOoIvFhyfWwKSkApfn2
+475QB0wD/rUci/bDj2TALdW0a0xRL0kFYq9JhJUy6rkvYUKjzLj0qzmVWdcQtpSwWjJrTds
22eJfG0Po6NL4+bQaC/qDJ1leK132DYeAWvgzjvP8iB7527swde67mvcfzALbJtXRaXpmlyL
qfzyzaVLkTEhgg6ffYDQpYSO7jgY1KF41Q+jPsqQp1/Kn7HfOlShPvDExfKlB6OHpQPu50om
oZXs27l7u6jwKhTbSd4bvDUXGyZWKHA=
============================ End of license ============================
        </body>
      </parse>
    </licensem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/licensem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="8c4794796bt6952r9b8" time="2008-01-21T14:21:00+0000">
<origin>licensem</origin>
<target>vzclient2-9634f0ee-8c54-924d-9193-64f79dfc8736</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <licensem>
    <license>
      <parameter>
        <name>CLASS</name>
        <value>VlpTUlY=</value>
      </parameter>
      <parameter>
        <name>owner_name</name>
        <value>RXZhbHVhdGlvbiBWejQuMCBzZXJ2ZXIgSQ==</value>
      </parameter>
      <parameter>
        <name>status</name>
        <value>VkFMSUQ=</value>
      </parameter>
      <parameter>
        <name>version</name>
```

```
        <value>NC4w</value>
      </parameter>
      <parameter>
        <name>owner_id</name>
        <value>MTEuMg==</value>
      </parameter>
      <parameter>
        <name>hwid</name>
        <value>MDAwMC4wMDAwLjAwMDAuMDAwMC4wMDAwLjAwMDAuMDAwMC4wMDAw</value>
      </parameter>
      <parameter>
        <name>serial</name>
        <value>NTc4RS43RjE3LkE5MEYuNzQyMS45RUU3LkRBOTIuMkVERC42ODI5</value>
      </parameter>
      <parameter>
        <name>expiration</name>
        <value>MjAwOC0wMi0wMVQwMDowMDowMCswMDAw</value>
      </parameter>
      <parameter>
        <name>start_date</name>
        <value>MjAwNi0wOS0wMVQwMDowMDowMCswMDAw</value>
      </parameter>
      <parameter>
        <name>issue_date</name>
        <value>MjAwNy0xMi0yN1QwNzowNjozMCswMDAw</value>
      </parameter>
      <parameter>
        <name>graceperiod</name>
        <value>MzAw</value>
      </parameter>
      <parameter>
        <name>key_number</name>
        <value>VlouMDAwMDAwMTMuMDAwNA==</value>
      </parameter>
      <parameter>
        <name>key_number_version</name>
        <value>MDAwNA==</value>
      </parameter>
      <parameter>
        <name>key_number_value</name>
        <value>MDAwMDAwMTM=</value>
      </parameter>
      <parameter>
        <name>product</name>
        <value>VmlydHVvenpv</value>
      </parameter>
      <parameter>
        <name>ct_total</name>
        <value>ODE5Mg==</value>
      </parameter>
      <parameter>
        <name>cpu_total</name>
        <value>NjQ=</value>
      </parameter>
      <parameter>
        <name>vzpp_allowed</name>
        <value>MQ==</value>
      </parameter>
      <parameter>
        <name>backup_mgmt_allowed</name>
        <value>MQ==</value>
      </parameter>
      <parameter>
        <name>workflow_mgmt_allowed</name>
        <value>MQ==</value>
      </parameter>
      <parameter>
```

```
        <name>vzagent_allowed</name>
        <value>MQ==</value>
      </parameter>
      <parameter>
        <name>max_vzmc_users</name>
        <value>MTI4</value>
      </parameter>
      <parameter>
        <name>max_vzcc_users</name>
        <value>MjU2</value>
      </parameter>
      <parameter>
        <name>architecture</name>
        <value>eDg2</value>
      </parameter>
      <parameter>
        <name>architecture</name>
        <value>eDg2XzY0</value>
      </parameter>
      <parameter>
        <name>architecture</name>
        <value>aWE2NA==</value>
      </parameter>
      <parameter>
        <name>platform</name>
        <value>YW55</value>
      </parameter>
    </license>
  </licensem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

delete

Summary:

Deletes a Virtuozzo Containers license from the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
delete	1..1		
{			
type	0..[]	string	This parameter is not currently used.
serial	0..[]	string	This parameter is not currently used.
}			

Returns:

OK/Error

Example:

```
<packet version="4.0.0">
  <target>licensem</target>
  <data>
    <licensem>
      <delete/>
    </licensem>
  </data>
</packet>
```

get_hwid

Summary:

Returns the Hardware Node ID associated with the installed Virtuozzo Containers license.

Request specification:

Name	Min/Max	Type	Description
get_hwid	1..1		

Returns:

Name	Min/Max	Type	Description
hwid	1..[]	string	Hardware Node ID.

Example:

Input

```
<packet version="4.0.0">
  <target>licensem</target>
  <data>
    <licensem>
      <get_hwid/>
    </licensem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/licensem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="9c47947c17t5f90r9b8" time="2008-01-21T14:28:09+0000">
<origin>licensem</origin>
<target>vzclient2-9634f0ee-8c54-924d-9193-64f79dfc8736</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <licensem>
    <hwid>0719.FD4A.CB5D.5681.8117.137A.DA24.AEE9</hwid>
  </licensem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

update

Summary:

Updates a Virtuozzo Containers license.

Request specification:

Name	Min/Max	Type	Description
`update`			
`{`			
`  serial`	`1..1`	`string`	A Base64-encoded string containing the license serial number.
`}`			

Returns:

Name	Min/Max	Type	Description
`license`	`1..1`	`licenseType (p. 273)`	The updated license information.

Description

The `update` call allows to update a Virtuozzo Containers license installed on a Hardware Node. The call will establish a connection with the SWsoft authentication server and will verify whether your existing license is eligible for the update. Upon approval, it will retrieve the new license information and will update your existing license.

Example:

```
<packet version="4.0.0">
  <target>licensem</target>
  <data>
    <licensem>
      <update>
        <serial>NTc4RS43RjE3LkE5MEYuNzQyMS45RUU3LkRBOTIuMkVERC42ODI5</serial>
      </update>
    </licensem>
  </data>
</packet>
```

mailer

Purpose:

The mail template configuration management interface.

Mail templates are used to build mail messages that are sent to the alert subscribers (p. 64). The mailer interface provides functionality that allows you to create your own mail templates. Typically, a resource allocation alert consists of the description of a problem, the type of the alert, the ID of the server that was affected, and the data related to the alert source. All this information can be included in a mail template using variables. You construct a template as a free-form text message and include the predefined variables in it where needed. The following is an example of a mail template:

```
You received this message as a subscriber of Virtuozzo alert
services. Message: $TYPE $ID alert on Container $TITLE Current
value: $CUR Soft limit: $SOFT Hard limit: $HARD
```

The variables (identified by the dollar sign in front of their names) will be substituted with the actual values at the time the message is generated. The following is a sample message that was composed using the template above:

```
You received this message as a subscriber of Virtuozzo alert services.
Message: red physpages alert on Container VC-101 Current value: 2147481270
Soft limit: 0 Hard limit: 2147483647
```

The following table describes the variables that you can use in your mail templates.

Variable	Description
`$TITLE`	Virtuozzo Container title.
`$TIME`	Time of the alert occurrence.
`$ID`	The name of a quota, ubc parameter, or service.

$TYPE	The alert type: green, red, yellow for quota alerts. running, stopped for service alerts.
$CUR	Current value (ubc or quota alerts).
$SOFT	Soft limit (ubc or quota alerts).
$HARD	Hard limit (ubc or quota alerts).

Types

Calls

Call	Description
mail_template_list (p. 288)	List all of the available mail templates.
get_mail_template (p. 289)	Gets mail template, specified by its name.
set_mail_template (p. 290)	Sets mail template.
remove_mail_template (p. 291)	Removes mail template.
set_relay (p. 292)	Sets smart relay for mail delivering.
get_relay (p. 293)	Gets smart relay.
post (p. 293)	Posts a new message.

mail_template_list

Summary:

Returns a list of the available mail templates.

Request specification:

Name	Min/Max	Type	Description
mail_template_list			

Returns:

Name	Min/Max	Type	Description
mail_tempate_list	0..1		A list of the templates.
{			
name	0..[]	string	Template name.
}			

Example:

Input

```
<packet version="4.0.0">
  <target>mailer</target>
  <data>
    <mailer>
      <mail_template_list/>
     </mailer>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
  <ns1:origin>mailer</ns1:origin>
  <ns1:target>vzclient3</ns1:target>
  <ns1:data>
    <ns2:mailer>
      <ns2:mail_template_list>
        <ns2:name>genericmail</ns2:name>
      </ns2:mail_template_list>
    </ns2:mailer>
  </ns1:data>
</ns1:packet>
```

get_mail_template

Summary:

Retrieves the specified mail template.

Request specification:

Name	Min/Max	Type	Description
get_mail_template			
{			
name	0..1	string	Template name.
}			

Returns:

Name	Min/Max	Type	Description
mail_template	0..1		Template info.
{			
name	1..1	string	Template name.
body	1..1	base64Binary	Template body.
read_only	1..1	none	If present then the template is read-only and cannot be modified.
}			

Example:

Input

```
<packet version="4.0.0">
  <target>mailer</target>
  <data>
    <mailer>
      <get_mail_template>
         <name>genericmail</name>
       </get_mail_template>
     </mailer>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>mailer</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:data>
<ns2:mailer>
<ns2:mail_template>
<ns2:name>genericmail</ns2:name>
<ns2:body>WW91IHJlY2VpdmVkIHRoaXMgbWVzc2FnZSBhcyBhIHN1YnNjcmliZXIgb2YgVmlydHVv
enpvIGFsZXJ0IHNlcnZpY2VzLiBNZXNzYWdlOiAkVFlQRSAkSUQgYWxlcnQgb24gRW52aXJvbm1lbn
QgJFRJVExFIEN1cnJlbnQgdmFsdWU6ICRDVVIgU29mdCBsaW1pdDogJFNPRlQgSGFyZCBsaW1pdDog
JEhBUkQ=</ns2:body>
</ns2:mail_template>
</ns2:mailer>
</ns1:data>
</ns1:packet>
```

set_mail_template

Summary:

Creates a mail template.

Request specification:

Name	Min/Max	Type	Description
set_mail_template			
{			
name	1..1	string	Template name.
body	1..1	base64Binary	Template body.
}			

Returns:

OK/Error.

Description:

For the details on how to compose a template body, see mailer (p. 287).

Example:

```
<packet>
<target>mailer</target>
<data>
<mailer>
<set_mail_template>
<name>genericmail</name>
<body>WW91IHJlY2VpdmVkIHRoaXMgbWVzc2FnZSBhcyBhIHN1YnNjcmliZXIgb2YgVmlydHVvenpv
IGFsZXJ0IHNlcnZpY2VzLiBNZXNzYWdlOiAkVFlQRSAkSUQgYWxlcnQgb24gRW52aXJvbm1lbnQgJF
RJVExFIEN1cnJlbnQgdmFsdWU6ICRDVVIgU29mdCBsaW1pdDogJFNPRlQgSGFyZCBsaW1pdDogJEhB
UkQ=</body>
</set_mail_template>
</mailer>
</data>
</packet>
```

remove_mail_template

Summary:

Removes existing mail template.

Request specification:

Name	Min/Max	Type	Description
`remove_mail_template`			
`{`			
`name`	`1..1`	`string`	Template name.
`}`			

Returns:

OK/Errors.

Example:

```
<packet version="4.0.0">
  <target>mailer</target>
  <data>
    <mailer>
      <remove_mail_template>
         <name>mytemplate</name>
       </remove_mail_template>
     </mailer>
  </data>
</packet>
```

set_relay

Summary:

Sets smart relay for mail delivering.

Request specification:

Name	Min/Max	Type	Description
set_relay			
{			
relay	1..1	ip_type (p. 26)	An IP address of a smart relay server. Only one smart relay server can be set. Each subsequent set_relay call overrides the previous setting.
}			

Returns:

OK/Error.

Example:

```
<packet version="4.0.0">
  <target>mailer</target>
  <data>
    <mailer>
      <set_relay>
        <relay>192.168.1.1</relay>
      </set_relay>
    </mailer>
  </data>
</packet>
```

get_relay

Summary:

Returns an IP address of the currently set smart relay server.

Request specification:

Name	Min/Max	Type	Description
get_relay	0..1		

Returns:

Name	Min/Max	Type	Description
relay	0..1	ip_type	An IP address of the smart relay server.

Example:

```
<packet version="4.0.0">
  <target>mailer</target>
  <data>
    <mailer>
      <set_relay>
        <relay>192.168.1.1</relay>
      </set_relay>
    </mailer>
  </data>
</packet>
```

post

Summary:

Posts a new mail message.

Request specification:

Name	Min/Max	Type	Description
post			
{			

message	1..1	base64Binary	Message. The data contained here will be fed to the sendmail program.
}			

Returns:

OK/Error.

Example:

Input

```
<packet>
  <target>mailer</target>
  <data>
    <mailer>
      <post>
        <!-- To: an_ur@yahoo.com From: an_ur@yahoo.com
        Subject:    mailc/post test Testing mailc/post -->
       <message>VG86IGFuX3VyQHlhaG9vLmNvbSANCkZyb206IGFuX3VyQH
       lhaG9vLmNvbSANClN1YmplY3Q6IG1haWxjL3Bvc3QgdGVz
       dA0KVGVzdGluZyBtYWlsYy9wb3N0</message>
       </post>
       </mailer>
       </data>
       </packet>
```

Output

```
<packet priority="0" version="4.0.0">
  <origin>mailer</origin>
  <target>vzclient2</target>
  <data>
    <mailer>
      <ok/>
    </mailer>
  </data>
</packet>
```

networkm

Purpose:

The `networkm` is a base network device management interface. The interface provides base functionality for managing network devices on the Hardware Node. The devices include network interfaces, network bridges, VLAN (virtual local area network) adapters. Supported virtualization technologies have their own network management interfaces, which are derived from the `networkm` interface.

Note: At the time of this writing, the only supported virtualization technology is Virtuozzo Containers.

Derived interfaces:

`vzanetworkm` (p. 606)

Types

net_vlanType

Summary:

Contains information about a VLAN adapter on the Hardware Node.

Type specification:

Extends `net_deviceType (p. 48)`

Adds the following elements:

Name	Min/Max	Type	Description
`vlan_id`	`1..1`	`string`	VLAN adapter ID (also called VLAN ID). The ID is used to identify a VLAN on the Hardware Node.
`base_device_id`	`1..1`	`string`	The name of the Physical network adapter this VLAN adapter is bound to.
`mac_address`	`0..1`	`string`	This is a read-only field. Contains the adapter's MAC address.

net_bridgeType

Summary:

Contains information about a network bridge on the Hardware Node.

Type specification:

Extends `net_deviceType (p. 48)`

The type has no additional elements.

Calls

Call	Description
add (p. 296)	Adds a network device.
list (p. 301)	Lists available network devices.
set (p. 307)	Sets network device parameters.
del (p. 311)	Removes a network device.

add

This call is available on Linux only.

Summary:

Adds a network device.

Request specification:

Name	Min/Max	Type	Description
`add`			
`{`			
`net_device`	`1..1`	`net_deviceType (p. 48)`	Network device information. Use the appropriate subtype of the `net_deviceType (p. 48)` based on the type of the device that you would like to add.
`}`			

Returns:

Name	Min/Max	Type	Description
net_device	0..[]	net_deviceType (p. 48)	Information about the newly created device.

Description:

To manage network devices for Virtuozzo-based systems, use the implementation of this call in the `vzaenvm` interface (p. 580).

This call allows to do the following:

- Create a network bridge on the Hardware Node.
- Add a VLAN adapter to the Hardware Node and connect it to a network bridge (to plug a physical network adapter into a bridge, use the `set` call (p. 307)).

Depending on the type of operation, use the appropriate subtype of the `net_deviceType (p. 48)` when populating the `net_device` parameter structure. For example, when adding a VLAN adapter, use `net_vlanType (p. 295)`; when adding a network bridge, use `net_bridgeType` (p. 296).

Creating a network bridge on the Hardware Node.

Before you can connect a virtual network adapter inside a Container to a physical or a virtual LAN, you first have to create a network bridge on the Hardware Node. The bridge will act as a binding interface between the adapter inside a Container and a physical or VLAN adapter on the Hardware Node.

To create a VLAN adapter, use `net_bridgeType` (p. 296) as the data type for the `net_device` element. The following explains how to populate the structure parameters:

`id` -- a unique name that you would like to use for the new bridge.

`network_id` -- a unique name identifying the Virtuozzo virtual network associated with this bridge. Again, you can use any name you prefer (e.g. `vznet1`). You will use this ID later to attach LAN/VLAN adapters and virtual network interfaces inside Containers to the bridge.

The `ip_address` and `dhcp` parameters are optional.

Example:

The following example shows how to create a bridge.

We use `vzbridge5` as the bridge name, and `vznet5` as the network ID.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <add>
        <net_device xsi:type="ns2:net_bridgeType">
          <ns2:id>vzbridge5</ns2:id>
          <ns2:network_id>dnpuZXQ1</ns2:network_id>
        </net_device>
      </add>
    </vzanetworkm>
  </data>
</packet>
```

Output

The output contains the information about the created network bridge, which indicates that the bridge was created successfully.

```
<ns1:packet xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="14c45c07df5t7e87rd5c" time="2007-01-31T09:14:17+0000" priority="0"
version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_bridgeType">
      <ns4:id>vzbridge5</ns4:id>
      <ns4:network_id>dnpuZXQ1</ns4:network_id>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Adding a VLAN adapter to the Hardware Node.

To create a VLAN adapter, use `net_vlanType (p. 295)` as the data type for the `net_device` element. The following explains how to populate the structure parameters.

Mandatory input parameters:

`vlan_id` -- the VLAN adapter ID. The ID is an arbitrary number that must be unique on this Hardware Node. The call will automatically generate a unique name for this adapter using this number. For example, if you specify 5 and attach the adapter to the `eth0` physical network adapter, the name of the VLAN adapter will be `eth0.5`. The name will be returned to the caller via the `id` element of the return structure.

`base_device_id` -- the name of the physical network adapter to attach the VLAN adapter to. This can be any physical network interface card installed on the Hardware Node. This association becomes permanent once the VLAN adapter is created. If later you decide to attach a VLAN adapter to a different physical adapter, you will have to delete and recreate it using the desired settings.

Optional input parameters:

`ip_address` -- assign an IP address (or multiple addresses) to the VLAN adapter if you would like the computers connected to it to be accessible from other networks.

`dhcp` -- a DHCP flag. Include this element in the request if you would like the VLAN adapter to receive the IP address through DHCP.

`network_id` -- Virtuozzo virtual network ID. Use this parameters if you would like to attach this VLAN adapter to a network bridge. A unique network ID is assigned to every Virtuozzo network bridge (see Creating a network bridge on the Hardware Node above).

Example:

The following example shows how to add a VLAN adapter to the Hardware Node running Linux OS. In the example, we are setting the device ID to 5, associating the VLAN adapter with the `eth1` physical network adapter, and attaching the adapter to the `vzbridge5` network bridge that we created earlier.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <add>
        <net_device xsi:type="ns2:net_vlanType">
          <ns2:vlan_id>5</ns2:vlan_id>
          <ns2:base_device_id>eth1</ns2:base_device_id>
          <ns2:network_id>dnpuZXQ1</ns2:network_id>
        </net_device>
      </add>
    </vzanetworkm>
  </data>
</packet>
```

Output

The output contains the information about the created VLAN adapter, including the ID that was generated automatically. In this case, the ID is `eth1.5`. As you can see, the ID is composed of the name of the physical interface this VLAN adapter is attached to (i.e. `eth1`) and the `vlan_id` that we specified in the request (i.e. `5`).

As a result, we have a new VLAN adapter created, and plugged into a network bridge. If you have virtual network adapters inside Containers connected to the bridge, they now have access to the VLAN adapter. If you don't have any virtual adapters connected to the bridge, or if you want to connect more, you may connect them at any time now.

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="15c45c0814bt390crd5c" time="2007-01-31T09:25:44+0000" priority="0"
version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_vlanType">
      <ns4:id>eth1.5</ns4:id>
      <ns4:network_id>dnpuZXQ1</ns4:network_id>
      <ns3:vlan_id>5</ns3:vlan_id>
      <ns3:base_device_id>eth1</ns3:base_device_id>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

list

Summary:

Lists available network devices.

Request specification:

Name	Min/Max	Type	Description
`list`			
`{`			
`net_device`	`0..[]`	`net_deviceType (p. 48)`	Used as a filter. Allows to search for a particular devices. To list all available devices, omit the element.
`}`			

Returns:

Name	Min/Max	Type	Description
net_device	0..[]	net_deviceType (p. 48)	A list of devices according to the specified criteria.

Description:

To manage network devices for Virtuozzo-based systems, use the implementation of this call in the `vzaenvm` interface (p. 580).

The call retrieves a list of network devices installed on the Hardware Node, including physical network adapters, VLAN adapters, and network bridges. To retrieve a complete list of all available network devices, include the empty `net_device` element. To retrieve the devices of a particular type, the devices belonging to a specific network, or a particular device by its ID or IP address, use the `net_device` input parameter to compose a filter.

Since Virtuozzo Containers doesn't support Virtuozzo VLAN adapters on Windows, most of the filter options are not used.

The following examples demonstrate how to compose a filter for the most common criteria, but you can filter on any of the fields of the input structure. Depending on the device type, use the appropriate subtype of the `net_deviceType (p. 48)` for the `net_device` input parameter.

Example 1:

To retrieve a list of all VLAN network adapters, specify the `net_vlanType (p. 295)` as the data type of the `net_device` element.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <list>
        <net_device xsi:type="ns2:net_vlanType"/>
      </list>
    </vzanetworkm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="17c45c08851t99rd5c"
time="2007-01-31T09:49:09+0000" priority="0" version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_vlanType">
      <ns4:id>eth1.1</ns4:id>
      <ns3:vlan_id>1</ns3:vlan_id>
      <ns3:base_device_id>eth1</ns3:base_device_id>
    </ns2:net_device>
```

```
    <ns2:net_device xsi:type="ns3:net_vlanType">
      <ns4:id>eth1.2</ns4:id>
      <ns4:network_id>dnpuZXQz</ns4:network_id>
      <ns3:vlan_id>2</ns3:vlan_id>
      <ns3:base_device_id>eth1</ns3:base_device_id>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_vlanType">
      <ns4:id>eth1.5</ns4:id>
      <ns4:network_id>dnpuZXQ1</ns4:network_id>
      <ns3:vlan_id>5</ns3:vlan_id>
      <ns3:base_device_id>eth1</ns3:base_device_id>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 2

To retrieve a list of all network bridges, specify the `net_bridgeType` (p. 296) as the data type of the `net_device` element.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <list>
        <net_device xsi:type="ns2:net_bridgeType"/>
      </list>
    </vzanetworkm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="18c45c0895dt124rd5c"
time="2007-01-31T09:52:44+0000" priority="0" version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_bridgeType">
      <ns4:id>vzbridge1</ns4:id>
      <ns4:network_id>vznetw==</ns4:network_id>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_bridgeType">
      <ns4:id>vzbridge2</ns4:id>
      <ns4:network_id>dnpuZXQx</ns4:network_id>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_bridgeType">
      <ns4:id>vzbridge3</ns4:id>
      <ns4:network_id>dnpuZXQz</ns4:network_id>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_bridgeType">
```

```
      <ns4:id>vzbridge5</ns4:id>
      <ns4:network_id>dnpuZXQ1</ns4:network_id>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 3

Retrieving all devices belonging to the specified network ID.

Input

```
<packet version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <list>
        <net_device>
          <network_id>dnpuZXQ1</network_id>
        </net_device>
      </list>
    </vzanetworkm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="19c45c08a1bt305erd5c" time="2007-01-31T09:55:15+0000" priority="0"
version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_bridgeType">
      <ns4:id>vzbridge5</ns4:id>
      <ns4:network_id>dnpuZXQ1</ns4:network_id>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_vlanType">
      <ns4:id>eth1.5</ns4:id>
      <ns4:network_id>dnpuZXQ1</ns4:network_id>
      <ns3:vlan_id>5</ns3:vlan_id>
      <ns3:base_device_id>eth1</ns3:base_device_id>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 4

To retrieve the list of the installed physical network adapters, specify the `net_nicType (p. 49)` as the data type of the `net_device` element.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <list>
        <net_device xsi:type="ns2:net_nicType"/>
      </list>
    </vzanetworkm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="22c45c76508t74dree0"
time="2007-02-04T08:37:28+0000" priority="0" version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_nicType">
      <ns3:id>eth0</ns3:id>
      <ns3:ip_address>
        <ns3:netmask>255.255.252.0</ns3:netmask>
        <ns3:ip>192.168.0.250</ns3:ip>
      </ns3:ip_address>
      <ns3:network_id>dnpuZXQ1</ns3:network_id>
      <ns3:mac_address>00:0c:29:60:4c:7f</ns3:mac_address>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_nicType">
      <ns3:id>eth1</ns3:id>
      <ns3:ip_address>
        <ns3:netmask>255.255.255.0</ns3:netmask>
        <ns3:ip>192.168.12.133</ns3:ip>
      </ns3:ip_address>
      <ns3:mac_address>00:0c:29:60:4c:89</ns3:mac_address>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 5

Retrieving a list of all available network devices.

Input

```
<packet version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
```

```
      <list>
        <net_device/>
      </list>
    </vzanetworkm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="39c45c84c6bt260dree0" time="2007-02-06T09:39:51+0000" priority="0"
version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient9</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:net_device xsi:type="ns3:net_nicType">
      <ns3:id>AMD PCNET Family PCI Ethernet Adapter #2</ns3:id>
      <ns3:mac_address>00:0c:29:23:76:30</ns3:mac_address>
    </ns2:net_device>
    <ns2:net_device xsi:type="ns3:net_nicType">
      <ns3:id>AMD PCNET Family PCI Ethernet Adapter - SWSoft network
bridge</ns3:id>
      <ns3:mac_address>00:0c:29:23:76:26</ns3:mac_address>
    </ns2:net_device>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

set

Summary:

Modifies the network device parameters.

Request specification:

Name	Min/Max	Type	Description
set			
{			
net_device	1..1	net_deviceType (p. 48)	The new device information.
}			

Returns:

OK/Error.

Description:

To manage network devices for Virtuozzo-based systems, use the implementation of this call in the `vzaenvm` interface (p. 580).

On Linux, the call allows to perform the following tasks:

- Attach/detach a LAN or VLAN adapter to/from a network bridge.
- Add/remove an IP address to/from a VLAN adapter.
- Set a VLAN adapter to use DHCP.

On Windows, the call allows to perform the following tasks:

- Assign a Virtuozzo virtual network ID to a physical network adapter or to a non-Virtuozzo virtual network.

Attaching/detaching LAN/VLAN adapters to/from a network bridge.

Available on Linux only.

First, create a network bridge (if you haven't done so already) using the `add` call (p. 296).

Use `net_vlanType (p. 295)` as a data type for the `net_device` element. To attach an adapter to a bridge, populate the following input parameters:

`id` -- the name of a physical LAN or VLAN adapter that you would like to plug into the bridge (e.g. `eth0`, `eth1.5`).

`network_id` -- a Virtuozzo virtual network ID that you would like to attach the network adapter to. If an adapter is already connected to a bridge and you pass a different network ID, the adapter will be disconnected from the old bridge and connected to the new bridge. To detach an adapter from a bridge, leave the `network_id` element empty.

`vlan_id` -- the VLAN adapter ID. When attaching/detaching a physical network adapter, leave the element empty (but still include it because it is mandatory according to the schema).

`base_device_id` -- the name of the physical network adapter with which the VLAN adapter is associated. When attaching/detaching a physical adapter, leave the element empty.

Example:

Connecting the physical network adapter `eht0` to the bridge `vznet6`.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <set>
        <net_device xsi:type="ns2:net_vlanType">
          <ns2:id>eth0</ns2:id>
          <ns2:network_id>dnpuZXQ2</ns2:network_id>
          <ns2:vlan_id/>
          <ns2:base_device_id/>
        </net_device>
      </set>
    </vzanetworkm>
  </data>
</packet>
```

Example 2

Connecting the VLAN adapter `eth1.5` to the bridge `vznet6`.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
```

```
    <vzanetworkm>
      <set>
        <net_device xsi:type="ns2:net_vlanType">
          <ns2:id>eth1.5</ns2:id>
          <ns2:network_id>dnpuZXQ2</ns2:network_id>
          <ns2:vlan_id>5</ns2:vlan_id>
          <ns2:base_device_id>eth1</ns2:base_device_id>
        </net_device>
      </set>
    </vzanetworkm>
  </data>
</packet>
```

Adding and removing an IP address to/from a VLAN adapter.

Available on Linux only.

If you include the `ip_address` element in this call, all existing IP addresses are deleted from the adapter first and then the specified addresses are added. To add an IP address, first retrieve the existing IP addresses using the `list` call and then compose the `ip_address` element to contain the existing and the new addresses (exclude the exciting IP addresses that you want to be removed from the adapter's configuration).

Assigning a Virtuozzo virtual network ID to a physical network adapter or to a non-Virtuozzo virtual network.

Available in Windows only.

Before you can set a virtual ethernet adapter inside a Container to use a specific physical network adapter or a non-Virtuozzo virtual network, you have to assign a Virtuozzo virtual network ID to that adapter or a network.

To retrieve the list of the available physical adapters and non-Virtuozzo virtual networks, use the `list` call (p. 301).

For the information on creating virtual adapters inside Virtuozzo Containers, see `net_vethType` (p. 569).

Example:

Assigning a network ID `vznet1` to the physical adapter `AMD PCNET Family PCI Ethernet Adapter #2`.

Input

```
<ns1:packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="32c45c847e6t63cbree0" time="2007-02-06T09:20:34+0000" priority="0"
version="4.0.0">
  <ns1:target>vzanetworkm</ns1:target>
  <ns1:data>
    <ns2:vzanetworkm>
      <ns2:set>
        <ns2:net_device xsi:type="ns3:net_nicType">
          <ns3:id>AMD PCNET Family PCI Ethernet Adapter #2</ns3:id>
          <ns3:network_id>dnpuZXQx</ns3:network_id>
        </ns2:net_device>
      </ns2:set>
    </ns2:vzanetworkm>
```

```
  </ns1:data>
</ns1:packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="38c45c84aeft2213ree0" time="2007-02-06T09:33:31+0000" priority="0"
version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient9</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns1:ok/>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

del

Summary:

Removes a device.

Request specification:

Name	Min/Max	Type	Description
`del`			
`{`			
`net_device`	`1..1`	`net_deviceType` (p. 48)	Device to remove.
`}`			

Returns:

OK/Error.

Description:

To manage network devices for Virtuozzo-based systems, use the implementation of this call in the `vzaenvm` interface (p. 580).

The call allows to remove VLAN adapters and network bridges from the Hardware Node that were added to it using the `add` call (p. 296) or the Virtuozzo `vznetcfg` command-line utility. To delete a device, specify its ID using the `id` element of the input structure. When deleting a VLAN adapter, use `net_vlanType (p. 295)` as a data type for the `net_device` input parameter. When deleting a bridge, use `net_bridgeType` (p. 296).

If a bridge has LAN or VLAN adapters plugged into it, the adapters will be disconnected from it and then the bridge will be deleted.

To retrieve a list of network devices from the Hardware Node, use the `list` call (p. 301).

Example 1:

Removing the `eth1.5` VLAN adapter.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <del>
        <net_device xsi:type="ns2:net_vlanType">
          <ns2:id>eth1.5</ns2:id>
        </net_device>
      </del>
    </vzanetworkm>
  </data>
</packet>
```

Example 2:

Removing the `vzbridge5` network bridge.

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/networkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <del>
        <net_device xsi:type="ns2:net_bridgeType">
          <ns2:id>vzbridge5</ns2:id>
        </net_device>
      </del>
    </vzanetworkm>
  </data>
</packet>
```

op_log

Purpose:

The operations log management interface. Provides access to the Agent history database.

Agent has a history database which is used for storing information about its operations on the Hardware Node or Virtuozzo Containers. To start the logging of an operation, you must specify the `log="on"` argument in the request message header. Once you initiate the logging, it will continue even if your client program disconnects from Agent. This, for example, allows you to start an operation, disconnect from the system, and review the results of the operation (including the intermediate results) later.

The logging is not available for all operations. In general, the operations that make any kind of modifications or additions (as opposed to the ones that simply retrieve the data) can be logged. The operations that normally take a long time to complete (such as backup for instance) can usually also be logged. The API calls that initiate an operation that support logging are marked in this guide using the *Logged Operation* annotation in the beginning of the section describing the call.

Calls

Call	Description
get_ops (p. 313)	Retrieves the information from a history database about currently running or completed operations.
put_ops (p. 316)	Saves a progress message into the Agent history database.

get_ops

Summary:

Retrieves information from the history database about currently running or completed operations.

Request specification:

Name	Min/Max	Type	Description
`get_ops`			
`{`			
`  eid`	`0..1`	`eid_type` (p. 25)	Server ID. If omitted, reports operations for all known servers.

status	0..1	none	If this element is included, only the initial progress message is returned for each operation along with the current status of the operation.
current	0..1	none	If this element is included, only the operations that are currently in progress are reported (just the initial progress message for each operation).
op	0..1	string	The name of the operation to get the report for. The name is comprised of the interface name and the call name separated by the dot character. For example, the name of the operation that creates a Virtuozzo Container is vzaenvm.create
id	0..1	string	The initial packet ID to get the report for.
start_time	0..1	datetime_type	Start time of the log.
end_time	0..1	datetime_type	End time of the log.
records	0..1	int	The number of records to retrieve. The records will be presented chronologically beginning with the most recent entry going back through time.
parent_id	0..1	none	Some operations are initiated not by the user directly but automatically by another (parent) operation. If this element is included, the parent operation ID for such operations will also be retrieved.
}			

Returns:

Name	Min/Max	Type	Description
progress	0..[]	progress_event Type (p. 565)	Progress information.

Example:

Input

```
<packet version="4.0.0">
  <target>op_log</target>
  <data>
    <op_log>
      <get_ops>
        <status/>
        <records>10</records>
      </get_ops>
    </op_log>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/op_log"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="3dc45f68aa5tbdbrdfc"
time="2007-03-11T06:04:56+0000" priority="0" version="4.0.0">
<ns1:origin>op_log</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns2:op_log>
    <ns2:progress>
      <ns2:id>13c45eee115t153crdfc</ns2:id>
      <ns2:op>vzaenvm.create</ns2:op>
      <ns2:message>

<ns2:message>T3BlcmF0aW9uICVvcF9uYW1lJSBpcyAlc3RhdHVzJQ==</ns2:message>
        <ns2:name></ns2:name>
        <ns2:translate/>
        <ns2:parameter>
          <ns2:message>Y3JlYXRl</ns2:message>
          <ns2:name>op_name</ns2:name>
        </ns2:parameter>
        <ns2:parameter>
          <ns2:message>c3RhcnRlZA==</ns2:message>
          <ns2:name>status</ns2:name>
          <ns2:translate/>
        </ns2:parameter>
      </ns2:message>
      <ns2:status>3</ns2:status>
      <ns2:time>2007-03-07T11:44:21+0000</ns2:time>
    </ns2:progress>
    <ns2:progress>
      <ns2:id>11c45eee0cat2ea6rdfc</ns2:id>
      <ns2:op>vzaenvm.create</ns2:op>
      <ns2:message>

<ns2:message>T3BlcmF0aW9uICVvcF9uYW1lJSBpcyAlc3RhdHVzJQ==</ns2:message>
        <ns2:name></ns2:name>
        <ns2:translate/>
        <ns2:parameter>
          <ns2:message>Y3JlYXRl</ns2:message>
          <ns2:name>op_name</ns2:name>
        </ns2:parameter>
        <ns2:parameter>
          <ns2:message>c3RhcnRlZA==</ns2:message>
          <ns2:name>status</ns2:name>
          <ns2:translate/>
        </ns2:parameter>
```

```
      </ns2:message>
      <ns2:status>4</ns2:status>
      <ns2:time>2007-03-07T11:43:31+0000</ns2:time>
    </ns2:progress>
  </ns2:op_log>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

put_ops

Summary:

Saves a progress message into the Agent history database.

Request specification:

Name	Min/Max	Type	Description
put_ops			
{			
progress	1..[]	progress_eventType (p. 565)	Progress data.
}			

Returns:

OK/Errors.

Description:

If you've received a progress message in real time from the operation that is not currently being logged in the history database, you can still save the message in the database manually using the `put_ops` call.

Example:

```
<packet version="4.0.0">
<target>op_log</target>
<data>
  <put_ops>
    <progress>
      <id>13c45eee115t153crdfc</id>
      <op>vzaenvm.create</op>
      <message>
        <message>T3BlcmF0aW9uICVvcF9uYW1lJSBpcyAlc3RhdHVzJQ==</message>
        <name></name>
        <translate/>
        <parameter>
          <message>Y3JlYXRl</message>
          <name>op_name</name>
        </parameter>
        <parameter>
          <message>c3RhcnRlZA==</message>
          <name>status</name>
          <translate/>
        </parameter>
      </message>
      <status>3</status>
      <time>2007-03-07T11:44:21+0000</time>
    </progress>
  </put_ops>
</data>
</packet>
```

packagem

Purpose:

The packagem interface provides calls for installing, updating, removing, and performing other operations on software packages on physical and virtual servers. Software packages may include templates (such as Virtuozzo templates), individual packages (rpm, deb), and others. Supported virtualization technologies have their own package management interfaces, which are derived from the packagem interface.

Derived interfaces:

vzapackagem (p. 612)

Types

package_debType

Summary:

Contains information about a Linux software package in Debian (deb) format.

Type specification:

Extends package_linuxType (p. 318)

The type has no additional elements.

package_linuxType

Summary:

Contains information about a generic Linux software package.

Type specification:

Extends packageType (p. 50)

Adds the following elements:

Name	Min/Max	Type	Description
path	0..1	base64Binary	Package location.

Subtypes:

package_rpmType (p. 319)

package_debType (p. 318)

package_rpmType

Summary:

Contains information about a Linux software package in RPM format.

Type specification:

Extends `package_linuxType (p. 318)`

The type has no additional elements.

packagesType

Summary:

Contains a list of software packages.

Type specification:

Name	Min/Max	Type	Description
`package`	`0..[]`	`packageType (p. 50)`	Software package information. Depending on the type of the package, the appropriate subtype of the `packageType (p. 50)` should/will be used here.

pkg_cmdType

Summary:

Defines the input parameters for some of the packagem interface (p. 318) calls.

Type specification:

Extends pkg_paramsType (p. 321)

Adds the following elements:

Name	Min/Max	Type	Description
eid	0..1	eid_type (p. 25)	Target Server ID. When removing a package, specifies the Server ID of a Virtuozzo Container from which to remove the package. When getting a list of packages or package information, specifies the Server ID of a Virtuozzo Container from which to get the list of installed packages. If this element is omitted, the Hardware Node is assumed to be the target. For example, the list of packages available on the Hardware Node will be retrieved.
packages	0..1	packagesType (p. 319)	Allows to specify package parameters that will be used as a filter when getting a list of packages or a package information. Only the packages with matching parameters will be included in the result. For example, you can use the filter to get the information for a particular package by specifying its name, or you can get the information for a particular package version, and so forth.

pkg_paramsType

Summary:

The base type defining the input parameters for the `packagem` interface calls.

Type specification:

The type has no elements.

Subtypes:

`pkg_cmdType` **(p. 320)**

`pkg_setup_cmdType` (p. 321)

pkg_setup_cmdType

Summary:

Defines the input parameters for some of the `packagem` interface calls. Contains the package setup information.

Type specification:

Extends `pkg_cmdType` **(p. 320)**

Adds the following elements:

Name	Min/Max	Type	Description
`eid`	`0..1`	`eid_type` (p. 25)	Target Server ID.
`installation_package`	`0..[]`		Package setup information.
`[`			*Denotes a choice between the* `package` *and the* `path` *elements.*
`package`	`1..1`	`packageType` (p. 50)	Package information.
`path`	`1..1`	`base64Binary`	Pathname specifying the package location.
`]`			
`options`	`0..1`	`none`	Package setup options.
`{`			

check	0..1	none	If this element is included, the call will run a simulation without actually installing a template or a package. You may include this option if you want to see the projected results of the installation, i.e the list of components (RPM packages) that will be added to the server.
force	0..1	none	Include this option to force the operation if possible. This may solve certain unexpected package installation problems. During the package update operation (p. 332), the option can be used to force the update even if the version installed is the latest one.
dependencies	0..1	none	This option is not currently used.
}			

Calls

Call	Description
install (p. 323)	Installs a package.
remove (p. 328)	Removes a package.
update (p. 332)	Updates a package.
list (p. 337)	Retrieves a list of installed packages.
get_info (p. 343)	Retrieves the specified package information.
clean (p. 346)	Cleans package manager cache.
fetch (p. 348)	Downloads EZ OS template packages from the remote repository to the local cache on the Hardware Node.
migrate (p. 351)	Migrates packages from one server to another.

install

Summary:

Installs an application or an operating system template on a Hardware Node or in a Virtuozzo Container. The call allows to install both standard and EZ Virtuozzo templates.

The call can also be used to install individual software packages (rpm, deb) on physical servers and in Virtuozzo Containers.

Request specification:

Name	Min/Max	Type	Description
`install`	`1..1`	`pkg_setup_cmdType` (p. 321)	The package installation parameters. Only the parameters from the `pkg_setup_cmdType` are used here. The parameters of the ancestor types (`pkg_cmdType`, `pkg_paramsType`) are not used. The `installation_package` element is mandatory and must contain the information about the package to be installed.

Returns:

Name	Min/Max	Type	Description
`packages`	`0..1`	`packagesType` (p. 319)	The results of the installation: For Virtuozzo OS template and application templates installed on the Hardware Node - the newly installed template information. For Virtuozzo application templates installed in a Container - the list of individual packages that were installed.

Installing an application or an OS template on the Hardware Node

Before an OS template can be used to create Virtuozzo Containers, or before an application template can be installed in a Virtuozzo Container, they must be installed on the Hardware Node. To install a template, the `path` parameter must contain the name and path of the template installation file (`.rpm` on Linux or `.efd` on Windows). The `eid` parameter can be omitted from the request or it must contain the Server ID of the Hardware Node. To install a template, the `vzapackagem` interface (p. 612) must be used.

The template installation on the Hardware Node consists of the following steps:

1 Installing (unpacking) the template package.

2 Caching the template.

Both steps are performed as a single operation (transparently to the user) using the `install` call described here.

EZ template specifics:

During an EZ template installation, the template data is downloaded from the Internet or from a local repository (the file download is a part of step 2 described above). In some situations, the download may fail for various reasons. In such a case, the template will still be installed on the Hardware Node (step 1) but will not be cached and, as a result, will not be valid. To remedy this situation, you may try using the `update` call (p. 332) to cache the template in a separate procedure. If caching of the template this way still results in error, you will need to contact Parallels technical support.

Example - Installing a standard operating system template:

The following sample illustrates how to install a standard operating system template on the Hardware Node. In the example, the `path` element contains the name and path of the template installation file (an rpm package in this instance). On operation completion, the template will be fully installed and cached, so you can use it immediately to create new Parallels Virtuozzo Containers.

Input

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <install>
        <installation_package>

<path>L3Z6L3Z6dXAyZGF0ZS90ZW1wbGF0ZXMvZmM1LnA0L3B1Yi9mZWRvcmEtY29yZTUtcDQtdG1w
bC0zLjAuMC0yLnN3c29mdC5pMzg2LnJwbQ==</path>
        </installation_package>
      </install>
    </vzapackagem>
  </data>
</packet>
```

Output

The output contains the information about the newly installed template.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" id="14c4863bd39t7e87r1be8" time="2008-06-26T19:48:52+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns2:package_std_vztemplateType">
        <name>fedora-core-5</name>
        <version>20070301</version>
        <summary>Fedora Core 5 OS Template</summary>
        <os xsi:type="ns3:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
        <arch>x86</arch>
        <os_template>1</os_template>
        <cached>0</cached>
        <uptodate>0</uptodate>
        <technology>nptl</technology>
        <technology>x86</technology>
        <base>0</base>
      </package>
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

Example - Installing an EZ operating system template:

Installing an EZ operating system template is similar to installing a standard OS template, except that the operation can take a significant amount of time since all of the template data must be downloaded from the Internet or from your local repository (to find out if you have a local repository, contact your system administrator). On successful operation completion, the template will be fully installed and cached, so you can use it to create new Virtuozzo Containers.

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <install>
        <installation_package>

<path>L3Z6L3Z6dXAyZGF0ZS9lenRlbXBsYXRlcy9mZWRvcmEtY29yZS03L3B1Yi9mZWRvcmEtY29y
ZS03LXg4Ni1lei0zLjAuMC0xMS5zd3NvZnQubm9hcmNoLnJwbQ==</path>
        </installation_package>
      </install>
    </vzapackagem>
  </data>
</packet>
```

Example - Installing a standard application template on the Hardware Node:

Installing a standard application template is no different from installing an OS template. All you have to do is specify the name and path of the template installation file and the `install` operation will take care of the rest. On success, the application template will be fully ready to be installed into Virtuozzo Containers.

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <install>
        <installation_package>

<path>L3Z6L3Z6dXAyZGF0ZS90ZW1wbGF0ZXMvZmM1LnA0L3B1Yi9wb3N0Z3Jlc3FsODE4LWZjNS10
bXBsLTMuMC4wLTIuc3dzb2Z0LmkzODYucnBt</path>
        </installation_package>
      </install>
    </vzapackagem>
  </data>
</packet>
```

Example - Installing an EZ application template on the Hardware Node:

The following example illustrates how to install an EZ application template on the Hardware Node. Please note that the actual files that comprise an EZ application template will not be downloaded during this operation. The files will be downloaded to the Hardware Node during the first-time installation of the template into a Virtuozzo Container.

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <install>
        <installation_package>

<path>L3Z6L3Z6dXAyZGF0ZS9lenRlbXBsYXRlcy9mZWRvcmEtY29yZS03L3B1Yi9teXNxbC1mZWRv
cmEtY29yZS03LXg4Ni1lei0zLjAuMC0xLnN3c29mdC5ub2FyY2gucnBt</path>
        </installation_package>
      </install>
    </vzapackagem>
  </data>
</packet>
```

Installing an Application template into a Virtuozzo Container

To install an application template into a Virtuozzo Container, the `eid` parameter must contain the Server ID of the Container. For a lists of the available templates, use the `list` call (p. 337). The data type of the `package` element must be the correct type for the package being installed. Please note that EZ templates are not compatible with standard templates. This means that you can only install an EZ application template into a Container that is based on the EZ OS template. The application template must also match the type and version of the operating system of the Container.

Note: Parallels Agent uses a dot notation to identify EZ templates. The dot character is placed in front of an EZ template name (e.g. `.mytemplate`). The same template name will show up without the dot character if you use the command line tools or the GUI tools. When referencing an EZ application template in Agent calls, always specify its name as it appears in the return of the `list` call (p. 337). When you create an EZ template yourself, make sure to begin the name of your template with the dot character as well.

Example - installing a standard application template into the specified Virtuozzo Container:

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <install>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <installation_package>
          <package xsi:type="ns1:package_std_vztemplateType">
             <ns1:name>postgresql-fc5</ns1:name>
          </package>
        </installation_package>
      </install>
    </vzapackagem>
  </data>
</packet>
```

Example - Installing an EZ application template into the specified Virtuozzo Container:

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <install>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <installation_package>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.mysql</ns1:name>
          </package>
        </installation_package>
      </install>
    </vzapackagem>
  </data>
</packet>
```

Output

On successful template installation, the output will contain the list of packages that were installed during this operation.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" id="33c4864b295t6bfcr1be8" time="2008-06-27T13:07:15+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
```

```
      <package xsi:type="ns1:package_rpmType">
        <name>mysql-bench</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql-devel</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql-server</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>perl-DBD-MySQL</name>
        <version>3.0007-1.fc6</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

Installing an RPM or DEB package on a Hardware Node

To install an RPM or DEB package on a Hardware Node, the `path` parameter must contain the name and the path of the package. The `eid` parameter must be omitted from the request or must contain the Server ID of the Hardware Node. Use the `packagem` interface (p. 318) to perform this operation. The data type of the `package` element must be the correct type for the package being installed.

remove

Summary:

Completely removes a template from the Hardware Node of a Virtuozzo Container. The call allows to remove both standard and EZ Virtuozzo templates.

Request specification:

Name	Min/Max	Type	Description
`remove`		`pkg_cmdType` **(p. 320)**	
{			

options	0..1		Removal options.
{			
check	0..1		Perform a test run without actually removing a package. The response message will contain the list of individual packages which will be removed as a result of this operation.
force	0..1		Force the operation in case of an error. This may solve certain unexpected problem during package removal.
dependencies	0..1		Remove all package dependencies.
}			
}			

Returns:

Name	Min/Max	Type	Description
packages	0..1	packagesType (p. 319)	The list of removed packages.

Description:

To remove an application template from a Virtuozzo Container, the `eid` parameter must contain the Server ID of the container. The `packages` structure must contain the information about the packages that you would like to remove.

To remove an application or an OS template from the Hardware Node, the `eid` parameter can be omitted from the request or must contain the Server ID of the Hardware Node. The `packages` parameter must contain the information about packages that you would like to remove. You can only remove templates that are not used by any of the Virtuozzo Containers installed on the Hardware Node.

EZ templates: If you would like to remove just the template cache (the actual data) but want to keep the template itself, use the `clean` call (p. 346) instead.

Example:

Removing a standard OS template from the Hardware Node.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <remove>
        <packages>
          <package xsi:type="ns1:package_std_vztemplateType">
             <ns1:name>fedora-core-5</ns1:name>
          </package>
        </packages>
      </remove>
    </vzapackagem>
  </data>
</packet>
```

Example:

Removing an EZ application template from the specified Virtuozzo Container.

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <remove>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <packages>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.mysql</ns1:name>
          </package>
        </packages>
      </remove>
    </vzapackagem>
  </data>
</packet>
```

Output

The output contains the list of individual packages that were removed from the Container.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" id="39c4864f0d1t260dr1be8" time="2008-06-27T17:32:46+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql-server</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql-devel</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>mysql-bench</name>
        <version>5.0.45-7.el5</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
      <package xsi:type="ns1:package_rpmType">
        <name>perl-DBD-MySQL</name>
        <version>3.0007-1.fc6</version>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
      </package>
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

update

Summary:

The update call provides functionality for updating Virtuozzo templates installed on the Hardware Node or in a Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
update		pkg_setup_cmdType (p. 321)	Template information and the update options. Different types of updates require different parameters and their values. Please see the descriptions and examples below.

Returns:

Name	Min/Max	Type	Description
packages	0..1	packagesType (p. 319)	A list of the templates that have been updated.

Description

The following types of updates can be performed using this call:

Standard templates:

- Caching all or a specific standard OS template. When installing a standard OS template for the first time, this step is automatically included in the `install` operation (p. 323). Situations when you would do this manually include recovery from a failed installation operation or any other situation when you have a standard OS template which was installed on the Hardware Node but was not cached.
- Installing a standard OS or an application template update in a Virtuozzo Container.

EZ templates:

- Caching all or a specific EZ OS template on the Hardware Node, which includes downloading the packages comprising a template and creating a template cache. When installing an EZ OS template for the first time, this step is automatically included in the `install` operation (p. 323). Situations when you would want do this manually include: caching a template after the `clean` operation (p. 346), recovery from a failed installation operation, or any other situation when you have an EZ OS template installed on the Hardware Node but not yet cached.
- Updating an OS template on the Hardware Node.
- Updating an OS or an application template in a Virtuozzo Container.

The following subsections describe each update operation and provide code samples.

Caching a standard template

To cache all of the templates available on the Hardware Node, pass an empty `update` element, as shown in the following example

Note: The call will cache all templates -- standard and EZ.

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update/>
    </vzapackagem>
  </data>
</packet>
```

To cache a specific OS template, supply the template name and version:

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <installation_package>
          <package xsi:type="ns1:package_std_vztemplateType">
             <ns1:name>fedora-core-5</ns1:name>
             <ns1:os_template>1</ns1:os_template>
             <ns1:version>20070301</ns1:version>
          </package>
        </installation_package>
      </update>
```

```
    </vzapackagem>
  </data>
</packet>
```

Installing a standard OS or an application template update in a Virtuozzo Container

To install a standard OS template update in a Virtuozzo Container, specify the Server ID (EID) of the Container and the name of the template containing the update (the update template must be installed on the Hardware Node (p. 323) prior to this operation). Please note that in order to install an update in a Virtuozzo Container, the base template must also be installed on the Hardware Node. The information about the required base template is included in the output of the `get_info` call (p. 343) for the template containing the update. For example, the following is a message describing the base template requirement for the `fedora-core-5` update:

```
<name>fedora-core-5</name>
<version>20070301</version>
<summary>Fedora Core 5 OS Template</summary>
<description>Fedora Core 5 packaged as a Virtuozzo/HSPcomplete template.
    Please note that this OS template is OS template update, so you should
have
    the following templates to be installed on Hardware Node before trying to
    install this OS template update:
    fedora-core5-p3-tmpl-3.0.0-1.swsoft
</description>
```

The following sample illustrates how to install an OS template update in the specified Virtuozzo Container.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <installation_package>
          <package xsi:type="ns1:package_std_vztemplateType">
             <ns1:name>fedora-core-5</ns1:name>
          </package>
        </installation_package>
      </update>
    </vzapackagem>
  </data>
</packet>
```

Caching all or a specific EZ OS template on the Hardware Node

To cache all installed, but not yet cached, EZ OS templates on the Hardware Node, pass an empty `update` element.

Note: The call will cache all templates -- standard and EZ.

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update/>
    </vzapackagem>
  </data>
</packet>
```

The following example will cache the EZ OS template with the specified name:

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <installation_package>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.fedora-core-7-x86</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </installation_package>
      </update>
    </vzapackagem>
  </data>
</packet>
```

Updating an EZ OS template on the Hardware Node

There are two kinds of EZ OS template updates:

- The update of the EZ template itself. This update comes as a new version of the template installation file. The update is installed and cached on the Hardware Node (both operations are performed in a single step). The packages comprising the template will be updated as a result.
- The update of the template cache. This update is performed using the existing template configuration (metadata). It is usually used when an update is available from the OS or the application vendor. This operation is exactly the same as the EZ template caching operation described earlier in this section. The difference is that in this case we are updating a template that has been cached already, but other than that the two operations are identical.

Both updates are installed using the `update` call described here. The following samples illustrate how each type of update is performed.

Installing an EZ template update:

The EZ template installation file must reside in a directory on the Hardware Node. Specify the name and path of the file using the `path` parameter. As a result of this operation, the corresponding EZ template will be updated using the instructions contained in the installation file.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <installation_package>

<path>L3Z6L3Z6dXAyZGF0ZS9lenRlbXBsYXRlcy9mZWRvcmEtY29yZS03L3B1Yi9mZWRvcmEtY29y
ZS03</path>
        </installation_package>
      </update>
    </vzapackagem>
  </data>
</packet>
```

Updating an EZ template cache

This update can be performed at any time and does not require any additional files or steps. All you have to do is specify the name of the template that you would like update. The operation will check if updates are available in the repository (local or remote, depending on your setup) and will install them if needed. Optionally, you may include the `force` element, in which case the template cache will be updated regardless if the updates are available or not.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <installation_package>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.fedora-core-7-x86</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </installation_package>
      </update>
    </vzapackagem>
  </data>
</packet>
```

Updating an EZ template in a Virtuozzo Container.

When you update an EZ template on the Hardware Node, the existing Containers that are already using it will not be affected. This means that they will continue using old versions of the template packages (the older packages are never deleted from the template cache). If you wish, you can update the Container to use the new packages.

The following sample illustrates how update a Container to use the newest packages from the specified OS template. The Container ID is specified using the eid parameter, the template name is specified using the name parameter.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <installation_package>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.redhat-el5-x86</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </installation_package>
      </update>
    </vzapackagem>
  </data>
</packet>
```

The following sample shows how to update a Container to use the latest packages from the specified application template (it is assumed that the application template is already installed in the Container).

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <update>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
```

```
        <installation_package>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.mysql</ns1:name>
             <ns1:os_template>0</ns1:os_template>
          </package>
        </installation_package>
      </update>
    </vzapackagem>
  </data>
</packet>
```

list

Summary:

The `list` call allows to obtain a list of Virtuozzo templates from the Hardware Node or a Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
`list`		`pkg_cmdType` **(p. 320)**	
`{`			
`options`			Listing options.
`{`			
`type`	`0..[]`	`string`	The type(s) of packages to retrieve. If absent, retrieves all types. The available options are: `os` -- Operating system template. `app` -- Application template. `rpm` -- Regular software package.
`summary`	`0..1`	`none`	If present, a summary info will be included in the result.
`compatible`	`0..1`	`none`	If present, only the packages that are compatible with the OS template used by the specified server will be included in the result.
`}`			
`}`			

Returns:

Name	Min/Max	Type	Description
`packages`	`0..1`	`packagesType (p. 319)`	The requested package information.

Description:

The following template information can be obtained using this call:

- List of Virtuozzo templates (standard/EZ, application/OS) installed on the Hardware Node.
- List of individual software packages comprising a template.
- Summary information about an OS template used by a Virtuozzo Container.
- List of application templates installed in a Virtuozzo Container.

Determining whether a returned template is a standart or an EZ Virtuozzo template

When examining the returned template list, the type of the template is determined based on the data type of the `package` element containing the template information. For example, the following XML code segment contains information about a Virtuozzo EZ template. We know that because the type of the `package` element is `package_vztemplateType (p. 572)`.

```
<ns2:package xsi:type="ns3:package_vztemplateType">
  <ns4:name>.redhat-el5-x86</ns4:name>
  <ns4:os xsi:type="ns4:osType">
    <ns4:platform>Linux</ns4:platform>
    <ns4:name/>
  </ns4:os>
  <ns4:arch>x86</ns4:arch>
  <ns3:os_template>1</ns3:os_template>
  <ns3:cached>0</ns3:cached>
  <ns3:uptodate>0</ns3:uptodate>
</ns2:package>
```

The following segment contains information about a standard template because the type of the `package` element in this case is `package_std_vztemplateType` (p. 572).

```
<ns2:package xsi:type="ns3:package_std_vztemplateType">
  <ns4:name>redhat-as3-minimal</ns4:name>
    <ns4:version>20061020</ns4:version>
    <ns4:os xsi:type="ns4:osType">
    <ns4:platform>Linux</ns4:platform>
  <ns4:name/>
  </ns4:os>
    <ns4:arch>x86</ns4:arch>
    <ns3:os_template>1</ns3:os_template>
    <ns3:cached>1</ns3:cached>
    <ns3:uptodate>0</ns3:uptodate>
    <ns3:technology>nptl</ns3:technology>
    <ns3:technology>x86</ns3:technology>
    <ns3:base>1</ns3:base>
  </ns2:package>
</ns2:packages>
```

EZ Template Names

Parallels Agent uses a dot notation to identify EZ templates. The dot character is placed in front of an EZ template name (e.g. `.redhat-el5-x86`). The same template name will show up without the dot character if you use the command line tools or the GUI tools. When referencing an EZ template in Agent calls, always use the dot notation. When you create an EZ application template yourself, make sure to begin the name of your template with the dot character as well. If you have created a Virtuozzo EZ template and installed it on the Hardware Node but don't see it in the list produced by this call, then it's probably because the name of your template does not have the dot character.

Obtaining a list of OS templates installed on the Hardware Node

The only required parameter here is template `type`. You can use the same call to retrieve a list of application templates from the Hardware Node by simply substituting the "os" value of the `type` parameter with "app".

Input

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <list>
        <options>
          <type>os</type>
        </options>
      </list>
    </vzapackagem>
  </data>
</packet>
```

Output

The output contains both standard and EZ templates.

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:ns3="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="3dc468a338bt440dre4"
time="2007-07-03T15:03:45+0000" priority="0" version="4.0.0">
<ns1:origin>vzapackagem</ns1:origin>
<ns1:target>vzclient5-b38361f0-6737-1342-9f9a-03b8fcca2130</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzapackagem>
    <ns2:packages>
      <ns2:package xsi:type="ns3:package_vztemplateType">
        <ns4:name>.redhat-el5-x86</ns4:name>
        <ns4:os xsi:type="ns4:osType">
          <ns4:platform>Linux</ns4:platform>
          <ns4:name/>
        </ns4:os>
        <ns4:arch>x86</ns4:arch>
        <ns3:os_template>1</ns3:os_template>
        <ns3:cached>0</ns3:cached>
        <ns3:uptodate>0</ns3:uptodate>
      </ns2:package>
      <ns2:package xsi:type="ns3:package_std_vztemplateType">
        <ns4:name>redhat-as3-minimal</ns4:name>
```

```
        <ns4:version>20061020</ns4:version>
        <ns4:os xsi:type="ns4:osType">
          <ns4:platform>Linux</ns4:platform>
          <ns4:name/>
        </ns4:os>
        <ns4:arch>x86</ns4:arch>
        <ns3:os_template>1</ns3:os_template>
        <ns3:cached>1</ns3:cached>
        <ns3:uptodate>0</ns3:uptodate>
        <ns3:technology>nptl</ns3:technology>
        <ns3:technology>x86</ns3:technology>
        <ns3:base>1</ns3:base>
      </ns2:package>
    </ns2:packages>
  </ns2:vzapackagem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Obtaining a list of individual packages comprising an OS template

To accomplish this goal, we have to include the following mandatory parameters:

`name` -- the OS template name.

`os_template` -- must contain `true` or `1` indicating that this is an OS template.

`version` -- the OS template version.

`options/type` -- must contain the "rpm" value, which indicates that we would like to get the information about packages included in the OS template, not the template itself.

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <list>
        <packages>
          <package xsi:type="ns1:package_std_vztemplateType">
             <ns1:name>fedora-core-5</ns1:name>
             <ns1:os_template>1</ns1:os_template>
             <ns1:version>20070301</ns1:version>
          </package>
        </packages>
        <options>
          <type>rpm</type>
        </options>
      </list>
    </vzapackagem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="6bc486b35d0t902r1be8" time="2008-07-02T11:29:07+0000">
<origin>vzapackagem</origin>
```

```
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns2:packageType">
        <name>audit-libs-1.3-2.fc5</name>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <version>20070301</version>
          <name>Fedora Core 5 OS Template</name>
        </os>
      </package>
      <package xsi:type="ns2:packageType">
        <name>audit-libs-python-1.3-2.fc5</name>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <version>20070301</version>
          <name>Fedora Core 5 OS Template</name>
        </os>
      </package>
      <package xsi:type="ns2:packageType">
        <name>bash-3.1-9.fc5.1</name>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <version>20070301</version>
          <name>Fedora Core 5 OS Template</name>
        </os>
      </package>
      <package xsi:type="ns2:packageType">
        <name>bind-9.3.4-1.fc5</name>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <version>20070301</version>
          <name>Fedora Core 5 OS Template</name>
        </os>
      </package>
      <package xsi:type="ns2:packageType">
        <name>bind-chroot-9.3.4-1.fc5</name>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <version>20070301</version>
          <name>Fedora Core 5 OS Template</name>
        </os>
      </package>
      <package xsi:type="ns2:packageType">
        <name>bind-libs-9.3.4-1.fc5</name>
        <os xsi:type="ns2:osType">
          <platform>Linux</platform>
          <version>20070301</version>
          <name>Fedora Core 5 OS Template</name>
        </os>
      </package>
      <!-- The rest of the output is omitted for brevity -->
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

Obtaining a list of individual packages comprising an application template

In this instance, the mandatory input parameters are:

`name` -- the application template name.

`os` -- a structure containing information about the OS template to which this application template belongs. The OS template information must include the `platform` (Linux, Windows, etc.) and the `name` (OS template name) parameters.

`options/type` -- must contain the "rpm" value, which indicates that we would like to get the information about packages included in the template, not the template itself.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <list>
        <packages>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.mysql</ns1:name>
             <ns1:os_template>0</ns1:os_template>
             <ns1:os>
               <platform>Linux</platform>
               <name>.fedora-core-7-x86</name>
             </ns1:os>
          </package>
        </packages>
        <options>
          <type>rpm</type>
        </options>
      </list>
    </vzapackagem>
  </data>
</packet>
```

Obtaining a list of application templates installed in the specified Virtuozzo Container

The parameters here are `eid` specifying the Server ID of the Container and the `type` specifying the template type. This call can also be used to obtain a summary information about an OS template used by the Container -- simply use the "os" value in the `type` element.

Input

```
<packet version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <list>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <options>
          <type>app</type>
        </options>
      </list>
    </vzapackagem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="41c4864f385t759ar1be8" time="2008-06-27T17:44:17+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns2:package_vztemplateType">
        <name>.mysql</name>
        <os xsi:type="ns3:osType">
          <platform>Linux</platform>
          <name>.redhat-el5-x86</name>
        </os>
        <os_template>0</os_template>
        <cached>0</cached>
        <uptodate>1</uptodate>
      </package>
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

get_info

Summary:

Returns information about a Virtuozzo template or an individual software package.

Request specification:

Name	Min/Max	Type	Description
`get_info`		`pkg_cmdType` **(p. 320)**	A list of packages to retrieve the info for. The `packages` parameter is mandatory and must contain at least `name` and `os_template` parameters.

Returns:

Name	Min/Max	Type	Description
`packages`	`0..1`	`packagesType (p. 319)`	The requested package information.

Description:

Compared to the `list` call (p. 337), which provides summary information about a template or a package, this call obtains a detailed package information. Use it when you want to retrieve complete information about a particular template or a package.

Example:

Retrieving information about a standard application template called `fedora-core-5` from the Hardware Node.

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <get_info>
        <packages>
          <package xsi:type="ns1:package_std_vztemplateType">
             <ns1:name>fedora-core-5</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </packages>
      </get_info>
    </vzapackagem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="45c4864f97dt5878r1be8" time="2008-06-27T18:09:44+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns2:package_std_vztemplateType">
        <name>fedora-core-5</name>
        <version>20070301</version>
        <summary>Fedora Core 5 OS Template</summary>
        <description>Fedora Core 5 packaged as a Virtuozzo/HSPcomplete
template.
          Please note that this OS template is OS template update, so you
should have
          the following templates to be installed on Hardware Node before
trying to
          install this OS template update:
          fedora-core5-p3-tmpl-3.0.0-1.swsoft</description>
        <os xsi:type="ns3:osType">
          <platform>Linux</platform>
          <name></name>
        </os>
        <arch>x86</arch>
        <os_template>1</os_template>
        <cached>0</cached>
```

```
        <uptodate>0</uptodate>
        <technology>nptl</technology>
        <technology>x86</technology>
        <base>0</base>
      </package>
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

Example:

Retrieving information about an EZ OS template called `.redhat-el5-x86`. Please note that in this case, we have to include the `os_template` element indicating the EZ template type (OS or application).

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <get_info>
        <packages>
          <package xsi:type="ns1:package_vztemplateType">
               <ns1:name>.redhat-el5-x86</ns1:name>
               <os_template>1</os_template>
          </package>
        </packages>
      </get_info>
    </vzapackagem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="56c4868bc0at1366r1be8" time="2008-06-30T14:34:53+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <packages>
      <package xsi:type="ns2:package_vztemplateType">
        <name>.redhat-el5-x86</name>
        <summary>Red Hat Enterprise Linux v. 5 Server EZ OS Template</summary>
        <description>Red Hat Enterprise Linux v. 5 Server packaged as a
Virtuozzo EZ Template.
        </description>
        <os xsi:type="ns3:osType">
          <platform>Linux</platform>
          <name>.redhat</name>
        </os>
        <arch>x86</arch>
        <os_template>1</os_template>
        <cached>1</cached>
```

```
        <uptodate>0</uptodate>

<path>L3Z6L3RlbXBsYXRlL3JlZGhhdC9lbDUveDg2L2NvbmZpZy9vcy9kZWZhdWx0</path>
        <technology>nptl</technology>
        <technology>x86</technology>
      </package>
    </packages>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

clean

Summary:

Clean repository metadata for the specified EZ template. This call can only be used with templates that are not currently installed in or used by any of the Virtuozzo Containers.

Request specification:

Name	Min/Max	Type	Description
`clean`			
`{`			
`packages`	`0..[]`	`packagesType` (p. 319)	A list of EZ templates to remove the data for.
`}`			

Returns:

OK/Errors.

Description:

This call removes the software packages, their headers, and metadata which were downloaded to the Hardware Node from the repository during the EZ template installation or update operation. The call does not remove the EZ template itself, so it will still appear in the list of the available templates with the status "not cached", which means that the template cannot be used to create new Containers (OS templates) or installed into a Container (application templates). Before you can use an OS template again, you have to cache it using the `update` call (p. 332). An application template will be re-deployed automatically as soon as you attempt to install it into a Container. Use this call when you want to free some hard drive space by removing the data of an unused EZ template without completely uninstalling it. Another usage scenario of this call is when you simply want to take a fresh version of the template data.

Example:

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <clean>
        <packages>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.fedora-core-7-x86</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </packages>
      </clean>
    </vzapackagem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="77c486b7576t4080r1be8" time="2008-07-02T16:00:42+0000">
<origin>vzapackagem</origin>
<target>vzclient2-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <ok/>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

fetch

Summary:

The `fetch` call is used to download packages included in the specified EZ OS template or their updates from the remote repository to the local cache on the Hardware Node and to prepare them for installation.

Request specification:

Name	Min/Max	Type	Description
`fetch`			
`{`			
`packages`	`1..1`	`packagesType` (p. 319)	EZ OS template information. Must include the template name.
`}`			

Returns:

OK/Errors.

Description:

The difference between the `fetch` call and the `install` or `update` call is that it downloads the packages but does not set up or update the template cache. You can run this operation as an unattended job during off-hours and use the downloaded data to set up the templates later. Please note that the Agent API does not provide functions to install a template from local cache. You will have to use the `vzpkg` command-line utility for that (see Parallels Virtuozzo Containers Reference Guide for more information).

Example:

The following sample will download packages for the `.fedora-core-7-x86` EZ OS template. To monitor the operation progress, we are including the `progress="on"` argument in the message header.

Input

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" progress="on"
version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <fetch>
        <packages>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.fedora-core-7-x86</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </packages>
      </fetch>
    </vzapackagem>
  </data>
</packet>
```

Output

The following is the initial progress message (there will be more than message like this).

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" type="1" time="2008-07-04T13:21:37+0000"
id="2c486df325t18ber114c">
<origin>vzapackagem</origin>
<target>vzclient5-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <progress>
    <op>vzapackagem.fetch</op>
    <message xsi:type="ns1:infoType">
      <message>ZmV0Y2ggcGFja2FnZXM6ICVwYWNrYWdlcyUgc3RhcnRlZA==</message>
      <name></name>
      <translate/>
      <parameter>
```

```
        <message>LmZlZG9yYS1jb3JlLTcteDg2</message>
        <name>packages</name>
        <translate/>
      </parameter>
      <parameter>
        <message>JXRpdGxlJQ==</message>
        <name>source_env</name>
        <translate/>
        <parameter>
          <message>NjdjZWZiNGEtOWE2ZS0yZDQxLTk0YWEtY2JmZDIwZmEzOTQz</message>
          <name>eid</name>
          <translate/>
        </parameter>
        <parameter>
          <message>bG9jYWxob3N0LmxvY2FsZG9tYWlu</message>
          <name>title</name>
          <translate/>
        </parameter>
        <parameter>
          <message>Z2VuZXJpYw==</message>
          <name>type</name>
          <translate/>
        </parameter>
      </parameter>
    </message>
    <status>1</status>
  </progress>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

On successful operation completion, the output will contain the standard "OK" message.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/packagem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="4000" id="2c486df325t18ber114c" time="2008-07-04T13:23:20+0000">
<origin>vzapackagem</origin>
<target>vzclient5-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <vzapackagem>
    <ok/>
  </vzapackagem>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

migrate

Summary:

Migrates a Virtuozzo template from one Hardware Node to another.

Request specification:

Name	Min/Max	Type	Description
migrate			
{			
packages	1..1	packagesType (p. 319)	Software packages to migrate.
src	0..1	connection_infoType (p. 30)	Source server connection information. When migrating a template from the current node, the parameter can be omitted.
dst	0..1	connection_infoType (p. 30)	Destination server connection information. When migrating a template from the remote node to the current node, the parameter can be omitted.
options	0..1	none	Migration options.
{			
policy	0..1	none	Policy.
{			
ignore_error	0..1	none	Use this option when migrating multiple packages. If this element is included, the call will ignore individual package errors and will continue with the next package in the list. If the element is omitted, the first error aborts the entire call.
}			
}			

Returns:

OK/Errors.

Description:

The call copies all the necessary template files from one node to another and sets the template up on the destination node. On completion, the new template on the destination node can be used to create Virtuozzo Containers (OS templates) or installed into a Container (application templates). The template on the source node will stay intact. When migrating an EZ OS template, only the template itself (the configuration files) will be copied to the destination node. After the copying is completed, the operation will automatically start caching of the template on the destination node. On success, the template will be fully ready to create Virtuozzo Containers on its basis.

Example:

The following example migrates an OS template called `.fedora-core-7-x86` from the current node to the node specified in the `dst` element.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" progress="on"
version="4.0.0">
  <target>vzapackagem</target>
  <data>
    <vzapackagem>
      <migrate>
        <packages>
          <package xsi:type="ns1:package_vztemplateType">
             <ns1:name>.fedora-core-7-x86</ns1:name>
             <ns1:os_template>1</ns1:os_template>
          </package>
        </packages>
        <dst>
          <address>10.30.25.145</address>
          <login>
            <name>cm9vdA==</name>
            <realm>00000000-0000-0000-0000-000000000000</realm>
          </login>
          <password>c2VjcmV0</password>
        </dst>
      </migrate>
    </vzapackagem>
  </data>
</packet>
```

perf_mon

Purpose:

The Performance Monitor interface.

Performance Monitor is an operator that allows to monitor performance of Hardware Nodes and Virtuozzo Containers. By monitoring the utilization of the system resources, you can acquire an important information about your system health. Performance Monitor can track a range of processes in real time and provide you with the results that can be used to identify current and potential problems. It can assist you with the tracking of the processes that need to be optimized, monitoring the results of the configuration changes, and identifying the resource usage bottlenecks.

The type of the resource and the aspect of the performance is specified by selecting a *class*, an *instance*, and a *counter* combination. The rest of this section explains what they are.

Performance Class is a type of the system resource that can be monitored, such as CPU, memory, disk, network. A class is identified by ID. See Appendix A: Performance Counters (p. 647) for a complete list of classes.

Class Instance refers to a particular device when multiple devices of the same type exist in the system. For example, your system may have multiple network interfaces or more than one hard disk drive. When monitoring a resource performance, you first have to select the appropriate class. The class alone, however, may not single out a particular resource when more than one device of the same type exist (e.g. multiple network interfaces). In such a case, you must also specify the instance that you would like to monitor. An instance is identified by the name of the device given to it by the operating system or Virtuozzo Containers. If only one instance of a resource exists, it is not necessary to supply the instance name. See Appendix A: Performance Counters (p. 647) provides information on how to obtain a list of instances for each class.

Performance Counters are used to measure various aspects of a performance, such as the CPU times, network rates, disk usage, etc. Each class has its own set of counters. The counter data is comprised of the current, minimum, maximum, and average values. For the complete list of performance counters see Appendix A: Performance Counters (p. 647).

Types

classType

Summary:

Contains performance class, instance, and counter information. See the `perf_mon` section (p. 352) for more information about classes, instances, and counters.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	The name of the class.

instance	0..[]		Class instances to monitor. If this element is omitted, the following will happen: ▪ all of the available instances of the class will be monitored. ▪ all of the counters from the specified class will be included in the report.
{			
name	0..1	string	Instance name.
counter	0..[]	string	Performance counters to include in the report. If this element is omitted, the report will contain all of the available counters from the specified class.
}			

compress_perf_dataType

Summary:

Type specification:

Name	Min/Max	Type	Description
data	0..[]		Performance data. Each instance of the data element will contain statistics for an individual server.
{			
eid	1..1	eid_type	Server ID.
value	1..1	base64Binary	
}			

Calls

Call	Description
start_monitor (p. 355)	Starts the performance monitor.
stop_monitor (p. 359)	Stops the performance monitor.
get (p. 360)	Obtains latest performance statistics for specified servers.
get_actual (p. 364)	Obtains latest performance statistics for all virtual servers running on the current physical server.

get_history (p. 364)	Obtains performance statistics for the specified time period for all virtual machines running on the current physical server.

start_monitor

Summary:

Use the `start_monitor` call to begin collecting performance data for the specified server(s).

Request specification:

Name	Min/Max	Type	Description
`start_monitor`			
`{`			
`eid_list`	`1..1`	`eid_listType (p. 32)`	A list containing Server IDs of the servers to monitor. Supply an empty list to monitor all known servers, including the Hardware Node and all of the Virtuozzo Containers that it hosts.
`filter`	`0..1`		Server filter options.
`{`			
`type`	`0..[]`	`string`	The type of the servers to monitor: `generic` -- will monitor just the Hardware Node. `virtuozzo` -- will monitor Virtuozzo Containers only.
`}`			

class	1..[]	classType (p. 353)	The list of the performance classes, instances, and counters to get the data for. You have to make sure that the classes and the counters specified here are compatible with all of the servers specified in the eid_list element and/or the type specified in the filter/type element. The following rules apply when selecting classes and counters: ▪ If the eid_list element contains just the Hardware Node or the type element is set to "generic", the classes must be of the generic type. ▪ If the list contains Virtuozzo Containers only or the type element is set to "virtuozzo", the classes must be of the virtuozzo type. ▪ If the eid_list element is empty and no filter is specified, you may mix classes and counters of both types -- the performance monitor will choose the correct classes and counters from the list for each server type automatically. ▪ If you mix a Hardware Node and Virtuozzo Containers in the same list, you have to make sure that each of the specified counters is compatible with both server types (i.e. a counter with the same name exists in both virtuozzo and generic lists). Failure to do so may give you unpredictable results. Normally, we don't recommend mixing servers of different types in the same request. If you really need to get performance data for the Hardware Node and Virtuozzo Containers in one call, use the scenario where an empty eid_list element is used (described above).

report_period	1..1	int	Reporting period in seconds. The collected data will be sent to the client at the time intervals specified here.
collect_period	0..1	int	This parameter is not currently used.
}			

Returns:

Monitor ID. This is the first response that you will receive from Agent. You will need this ID to stop the monitor later.

Name	Min/Max	Type	Description
id	1..1	guid_type (p. 26)	Monitor ID.

The collected performance data. The data will be sent to the client at the time intervals specified in the request.

Name	Min/Max	Type	Description
data	1..[]	perf_dataType (p. 51)	Performance data.

Description:

To begin collecting performance data, select the performance classes and counters, the servers for which the data should be collected, and the time intervals at which you would like to receive performance reports. See `perf_mon` (p. 352) for more information on classes, instances, and counters. To stop the monitor, use the `stop_monitor` call (p. 359).

Example:

The following example shows how to monitor CPU usage by the specified Virtuozzo Container.

Input

```
<packet version="4.0.0">
  <target>perf_mon</target>
  <data>
    <perf_mon>
      <start_monitor>
        <eid_list>
          <eid>39f40723-b3f5-8c41-8de9-7beefd5021fe</eid>
        </eid_list>
        <class>
          <name>counters_vz_cpu</name>
          <instance>
            <counter>counter_cpu_system</counter>
          </instance>
        </class>
        <report_period>20</report_period>
      </start_monitor>
    </perf_mon>
  </data>
</packet>
```

Output

The first response that we receive from Agent contains the monitor ID. We will need this ID to stop the monitor later.

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/perf_mon"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="bc470a4258t4ae1re38"
time="2007-10-08T16:35:45+0000" priority="0" version="4.0.0">
<origin>perf_mon</origin>
<target>vzclient2-ac6ab656-8558-0949-a605-f47cfc63cd9c</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <perf_mon>
    <id>e9cd9b4e-a386-9f42-84f7-4d1baae4e4b7</id>
  </perf_mon>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

The subsequent responses will contain the collected performance data.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vza/3.0.3/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="bc470a4258t4ae1re38"
time="2007-10-08T16:37:05+0000" priority="0" version="4.0.0">
```

```
<origin>perf_mon</origin>
<target>vzclient2-ac6ab656-8558-0949-a605-f47cfc63cd9c</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <perf_mon>
    <data xsi:type="ns1:perf_dataType">
      <eid>39f40723-b3f5-8c41-8de9-7beefd5021fe</eid>
      <interval xsi:type="ns2:intervalType">
        <start_time>2007-10-08T16:36:36+0000</start_time>
        <end_time>2007-10-08T16:36:56+0000</end_time>
      </interval>
      <class>
        <name>counters_vz_cpu</name>
        <instance>
          <name></name>
          <counter>
            <name>counter_cpu_system</name>
            <value>
              <avg>68</avg>
              <min>68</min>
              <max>68</max>
              <cur>68</cur>
            </value>
          </counter>
        </instance>
      </class>
    </data>
  </perf_mon>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

stop_monitor

Summary:

Stops the specified performance monitor instance.

Request specification:

Name	Min/Max	Type	Description
`stop_monitor`			
{			
id	1..1	`guid_type` (p. 26)	Monitor ID.
}			

Description:

The call stops the specified performance monitor. You receive the monitor ID from Agent when you start the monitor with the `start_monitor` call (p. 365).

Example:

```
<packet version="4.0.0" id="2">
  <target>perf_mon</target>
  <data>
    <perf_mon>
      <stop_monitor>
        <id>f4315d31-c017-4362-a213-4b0ea76860f3</id>
      </stop_monitor>
    </perf_mon>
  </data>
</packet>
```

get

Summary:

Obtains latest performance statistics for specified servers.

Request specification:

Name	Min/Max	Type	Description
`get`			
`{`			
`eid_list`	`1..1`	`eid_listType` (p. 32)	A list containing the IDs of the servers for which to retrieve the performance data. If this element is empty, the data for all known servers will be retrieved, including the Hardware Node and all of the Virtuozzo Containers that it hosts.

class	1..[]	classType (p. 353)	The list of the performance classes, instances, and counters to get the data for. You have to make sure that the classes and the counters specified here are compatible with the servers specified in the eid_list element. The following rules apply when selecting classes and counters: ▪ If the eid_list element contains just the Hardware Node, the classes must be of the generic type. ▪ If the list contains Virtuozzo Containers only, the classes must be of the virtuozzo type. ▪ If the eid_list element is empty, you may mix classes and counters of both types -- the performance monitor will choose the correct classes and the counters from the list for each server type automatically. ▪ If you mix a Hardware Node and Virtuozzo Containers in the same list, you have to make sure that each of the specified counters is compatible with both server types (i.e. a counter with this name exists in both virtuozzo and generic lists). Failure to do so may give you unpredictable output. Normally, we don't recommend mixing servers of different types in the same request. If you really need to get performance data for the Hardware Node and Virtuozzo Containers in one call, use the scenario where an empty eid_list element is used (described above).
}			

Returns:

Name	Min/Max	Type	Description
`data`	`1..[]`	`perf_dataType` (p. 51)	Performance data. Each instance of the `data` element will contain statistics for an individual server.

Description:

The performance data is collected by Agent for all running servers at the predefined time intervals (a few seconds) and is stored in a temporary storage buffer. The `get` call allows to retrieve the most recently collected data. This is an on-demand request. It produces a single response containing the latest performance data. If you would like to receive the performance reports on a periodic basis, use the `start_monitor` (p. 355) call instead. If you are using the Parallels Agent SOAP API, use this call to obtain performance statistics.

Please also see the `perf_mon` section (p. 352) for more information on classes, instances, and counters.

Example:

The following example shows how to get the latest performance data for the specified server using the specified class and counter.

Input

```
<packet version="4.0.0" id="2">
  <target>perf_mon</target>
  <data>
    <perf_mon>
      <get>
        <eid_list>
          <eid>1649d32f-3e18-6642-9690-4fe3cd406eb0</eid>
        </eid_list>
        <class>
          <name>counters_vz_cpu</name>
          <instance>
            <counter>counter_cpu_system</counter>
          </instance>
        </class>
      </get>
    </perf_mon>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/perf_mon"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="3fc4683bf1bt54derb2c" time="2007-06-28T17:39:58+0000" priority="0"
version="4.0.0">
<ns1:origin>perf_mon</ns1:origin>
<ns1:target>vzclient1-6d9ea6b6-e470-424b-98ca-27dd10e49860</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:perf_mon>
    <ns2:data>
      <ns2:eid>1649d32f-3e18-6642-9690-4fe3cd406eb0</ns2:eid>
      <ns2:interval>
        <ns2:start_time>2007-06-28T17:21:59+0000</ns2:start_time>
        <ns2:end_time>2007-06-28T17:39:39+0000</ns2:end_time>
      </ns2:interval>
      <ns2:class>
        <ns2:name>counters_vz_cpu</ns2:name>
        <ns2:instance>
```

```
          <ns2:name></ns2:name>
          <ns2:counter>
            <ns2:name>counter_cpu_system</ns2:name>
            <ns2:value>
              <ns2:avg>215</ns2:avg>
              <ns2:min>215</ns2:min>
              <ns2:max>215</ns2:max>
              <ns2:cur>215</ns2:cur>
            </ns2:value>
          </ns2:counter>
        </ns2:instance>
      </ns2:class>
    </ns2:data>
  </ns2:perf_mon>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_actual

Summary:

Obtains latest performance statistics for all virtual servers running on the current physical server.

Request specification:

The call has no parameters.

Returns:

Name	Min/Max	Type	Description
`data`	`1..1`	`compress_perf_dataType (p. 354)`	Performance data.

get_history

Summary:

Obtains performance statistics for the specified time period for all virtual machines running on the current physical server.

Request specification:

Name	Min/Max	Type	Description
`get_history`			
`{`			
`  interval`	`1..1`	`intervalType (p. 42)`	Time period for which to obtain statistics.
`}`			

Returns:

Name	Min/Max	Type	Description
data	1..1	compress_perf_dataType (p. 354)	Performance data.

proc_info

Purpose:

The base process monitoring interface. The interface allows to monitor system processes on Hardware Nodes and Virtuozzo Containers. Supported virtualization technologies have their own process monitoring interfaces, which are derived from the proc_info interface.

Note: At the time of this writing, the only supported virtualization technology is Virtuozzo Containers.

Derived interfaces:

vzaproc_info (p. 612).

Calls

Calls

Call	Description
start_monitor (p. 365)	Starts the monitor.
stop_monitor (p. 370)	Stops the monitor.
get (p. 370)	Retrieves a list of running processes.

start_monitor

Summary:

Starts the process monitor.

Request specification:

Name	Min/Max	Type	Description
start_monitor			
{			
period	1..1	int	Reporting period in seconds.

key	0..1	string	Parameter to order the result set by. See Description below for the list of parameters.
limit	0..1	int	Maximum number of processes to include in the report.
descending	0..1		If present, the list will be ordered in descending order.
}			

Returns:

Name	Min/Max	Type	Description
ps_info	1..1	ps_infoType (p. 53)	Processes information.

Description:

The call starts the process monitor on the server. The monitor sends the collected data back to the client at the specified time intervals until the client stops the monitor (p. 370) or disconnects from Agent. Only one process monitor can be running for a given connection.

The following table lists the parameters that can be specified in the key element to sort the result set by.

Parameter	Type	Description
pid	int	The process ID.
user	string	The user who has launched the process.
pri	int	The kernel scheduling priority for the process.
ni	int	The 'nice' parameter value defining the overall scheduling priority for the process. The less the 'nice' value, the higher the process priority.
rss	int	The total amount of physical memory used by the process, in kilobytes.
stat	string	The process current status. Can be 'R' (running), 'S' (sleeping, waiting for 'wake-up call'), 'D' (uninterruptable sleep), 'Z' (zombie, waiting for parent process), 'T' (stopped or traced). Sometimes the second symbol may appear: 'W' (process swapping), 'N' ('nice' process), 'L' (process has pages locked into memory). If the < sequence is displayed after the status, it means that this information was returned by the Parallels Agent software which, in turn, got this information from the 'ps' tool.
%cpu	float	The CPU time, in percent, used by the process.
%mem	float	The amount of physical memory, in megabytes, used by the process.
time	string	The total CPU time the process has used since its launch.

command	string	The command that invoked the process.

Example:

Input

```
<packet version="4.0.0">
  <target>proc_info</target>
  <data>
    <proc_info>
      <start_monitor>
        <period>10</period>
        <key>%cpu</key>
        <limit>5</limit>
      </start_monitor>
    </proc_info>
  </data>
</packet>
```

Output

The very first response contains the monitor ID an indicates that the monitor has been started.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/proc_info"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="7c487726act72aera18" time="2008-07-11T12:28:20+0000">
<origin>proc_info</origin>
<target>vzclient7-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <proc_info>
    <id>d2c2330e-33c2-1946-93b3-ae7c1f10650c</id>
  </proc_info>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

The subsequent responses will contain the collected process information.

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" time="2008-07-11T12:32:35+0000" id="1c487727act4823r175c">
<origin>proc_info</origin>
<target>vzclient8-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <proc_info>
    <ps_info xsi:type="ns1:ps_infoType">
      <process>
        <pid>4</pid>
        <param>MC4w</param>
        <param>MC4w</param>
        <param>IFtldmVudHMvMF0=</param>
        <param>LTU=</param>
        <param>Mjk=</param>
        <param>MA==</param>
        <param>Uzw=</param>
        <param>MDA6MDA6MDM=</param>
        <param>cm9vdA==</param>
      </process>
```

```
      <process>
        <pid>5</pid>
        <param>MC4w</param>
        <param>MC4w</param>
        <param>IFtraGVscGVyXQ==</param>
        <param>LTU=</param>
        <param>Mjk=</param>
        <param>MA==</param>
        <param>Uzw=</param>
        <param>MDA6MDA6MDA=</param>
        <param>cm9vdA==</param>
      </process>
      <process>
        <pid>2</pid>
        <param>MC4w</param>
        <param>MC4w</param>
        <param>IFttaWdyYXRpb24vMC8wXQ==</param>
        <param>LQ==</param>
        <param>MTM5</param>
        <param>MA==</param>
        <param>Uw==</param>
        <param>MDA6MDA6MDA=</param>
        <param>cm9vdA==</param>
      </process>
      <process>
        <pid>1</pid>
        <param>MC4w</param>
        <param>MC4x</param>

<param>IGluaXQgWzNdICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg</param>
        <param>MA==</param>
        <param>MjQ=</param>
        <param>NjMy</param>
        <param>U3M=</param>
        <param>MDA6MDA6MDM=</param>
        <param>cm9vdA==</param>
      </process>
      <process>
        <pid>3</pid>
        <param>MC4w</param>
        <param>MC4w</param>
        <param>IFtrc29mdGlycWQvMF0=</param>
        <param>MTk=</param>
        <param>MA==</param>
        <param>MA==</param>
        <param>U04=</param>
        <param>MDA6MDA6MDA=</param>
        <param>cm9vdA==</param>
      </process>
      <param_id>%cpu</param_id>
      <param_id>%mem</param_id>
      <param_id>command</param_id>
      <param_id>ni</param_id>
      <param_id>pri</param_id>
      <param_id>rss</param_id>
      <param_id>stat</param_id>
      <param_id>time</param_id>
      <param_id>user</param_id>
      <run>0</run>
      <zombie>0</zombie>
      <sleep>0</sleep>
      <uninterrupt>0</uninterrupt>
      <stopped>0</stopped>
      <idle>0</idle>
      <total>193</total>
    </ps_info>
```

```
  </proc_info>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

stop_monitor

Summary:

Stops the process monitor.

Request specification:

Name	Min/Max	Type	Description
stop_monitor	1..1	none	

Description:

The call stops the process monitor that was started previously with the `start_monitor` call (p. 365).

Example:

```
<packet version="4.0.0">
  <target>proc_info</target>
  <data>
    <proc_info>
      <stop_monitor/>
    </proc_info>
  </data>
</packet>
```

get

Summary:

Retrieves a list of processes running on a server.

Request specification:

Name	Min/Max	Type	Description
get	1..1		
{			
key	0..1	string	Parameter to order the resulting list of processes by. See the `start_monitor` call (p. 365) for the list of parameters.
limit	0..1	int	Maximum number of processes to include in the list.
descending	0..1	none	If present, the resulting list will be sorted in descending order.
}			

Returns:

Name	Min/Max	Type	Description
ps_info	0..1	ps_infoType (p. 53)	The list of processes.

Description:

This is a synchronous operation that returns a single report about processes running on the specified server. For an equivalent asynchronous operation see the `start_monitor` call (p. 355).

Example:

Input

```
<packet version="4.0.0">
  <target>proc_info</target>
  <data>
    <proc_info>
      <get>
        <key>%cpu</key>
        <limit>5</limit>
      </get>
    </proc_info>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/proc_info"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="4c48772d22t4ae1r175c" time="2008-07-11T12:55:54+0000">
<origin>proc_info</origin>
<target>vzclient8-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <proc_info>
    <ps_info xsi:type="ns2:ps_infoType">
      <process>
        <pid>4</pid>
        <param>MC4w</param>
        <param>MC4w</param>
        <param>IFtldmVudHMvMF0=</param>
        <param>LTU=</param>
        <param>Mjk=</param>
        <param>MA==</param>
        <param>Uzw=</param>
        <param>MDA6MDA6MDM=</param>
        <param>cm9vdA==</param>
      </process>
      <process>
        <pid>5</pid>
        <param>MC4w</param>
        <param>MC4w</param>
        <param>IFtraGVscGVyXQ==</param>
        <param>LTU=</param>
        <param>Mjk=</param>
        <param>MA==</param>
        <param>Uzw=</param>
        <param>MDA6MDA6MDA=</param>
        <param>cm9vdA==</param>
      </process>
      <process>
        <pid>2</pid>
        <param>MC4w</param>
        <param>MC4w</param>
```

```
        <param>IFttaWdyYXRpb24vMC8wXQ==</param>
        <param>LQ==</param>
        <param>MTM5</param>
        <param>MA==</param>
        <param>Uw==</param>
        <param>MDA6MDA6MDA=</param>
        <param>cm9vdA==</param>
      </process>
      <process>
        <pid>1</pid>
        <param>MC4w</param>
        <param>MC4x</param>

<param>IGluaXQgWzNdICAgICAgICAg
```

processm

Purpose:

The system processes management interface.

Calls

Call	Description
execute (p. 374)	Executes a program inside an server.
kill (p. 377)	Send a signal to the specified process.

execute

Summary:

Executes a program on a server.

Request specification:

Name	Min/Max	Type	Description
`execute`			
`{`			
`argv`	`0..[]`	`base64Binary`	Program name and expected arguments in proper order.
`envp`	`0..[]`	`base64Binary`	Definitions of the OS environment variables used by the program (if any).
`cred`	`1..1`		Execute the program as user specified here. By default, the default administrative account will be used (e.g. `root`).
`{`			
`[`			Specify either the user name or the user ID.
`user`	`0..1`	`string`	User name.
`uid`	`0..1`	`long`	User ID.
`]`			
`[`			Specify either the group name or the group ID.

group	0..[]	string	Group name. You may specify more than one group. If you do, the first group will be set as the effective group, the others will be set as supplementary groups. The supplementary group list will affect all of the new elements.
gid	0..[]	long	Group id. NOTE: There are no dedicated operations to get a group ID by its name. Therefore, do not use this parameter if you're not sure it is correct. If it is wrong, the entire operation will be canceled.
]			
chroot	0..1	base64Binary	Run command or interface shell with root directory set to the value specified here (same as chroot Linux command).
chdir	0..1	base64Binary	Change working directory.
umask	0..1	int	Operation umask.
}			
stdio	0..1		Standard input/output options.
{			
input	0..1	base64Binary	Program input.
output	0..1	none	Include this element to report standard program output.
error	0..1	none	Include this element to report program error output.
mixed	0..1	none	Include this element to report mixed output.
}			
}			

Returns:

Name	Min/Max	Type	Description
exec	0..1		The program execution results.
{			
status	1..1	int	Execution status. Will contain 0 (zero) on success or non-zero value on error.
output	0..1	base64Binary	Program standard output.
error	0..1	base64Binary	Program error output.
}			

Example:

Executing the `ls` command inside a Linux-based Virtuozzo Container.

Input

```
<packet version="4.0.0">
  <dst>
    Host37ea79c2-3565-0e43-887f-91d4678c843e</host>
  </dst>
  <target>processm</target>
  <data>
    <processm>
      <execute>
        <argv>bHM=</argv>
        <cred>
          <chroot>/</chroot>
        </cred>
        <stdio>
          <output/>
          <error/>
        </stdio>
      </execute>
    </processm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/processm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="4fc4631f282t2350re14" time="2007-04-27T00:55:52+0000" priority="0"
version="4.0.0">
<ns1:origin>37ea79c2-3565-0e43-887f-91d4678c843e</ns1:origin>
<ns1:dst>
  <ns1:director>gend</ns1:director>
  <ns1:target>vzclient3</ns1:target>
</ns1:dst>
<ns1:data>
  <ns2:processm>
    <ns2:exec>
      <ns2:status>0</ns2:status>

<ns2:output>YmluCmJvb3QKZGV2CmV0Ywpob21lCmluaXRyZApsaWIKbW50Cm9wdApwcm9jCnJvb3
QKc2Jpbgp0bXAKdXNyCnZhcgp+dnphZ2VudDAK</ns2:output>
      <ns2:error></ns2:error>
    </ns2:exec>
  </ns2:processm>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>processm</ns1:target>
</ns1:src>
<ns1:target>vzclient3</ns1:target>
</ns1:packet>
```

If you decode the data contained in the `output` element, you'll get the requested root directory listing:

```
bin
boot
dev
etc
home
```

```
initrd
lib
mnt
opt
proc
root
sbin
tmp
usr
var
```

kill

Summary:

Send a signal to the specified process. This call works only with Hardware Node. For a Virtuozzo Containers implementation, see `vzaprocessm/kill` (p. 620).

Request specification:

Name	Min/Max	Type	Description
kill			
{			
pid	1..[]	int	Process ID.
signal	1..1	int	Signal number.
}			

Returns:

OK/Error

Description:

The call sends a signal to a process on a server. The signal number can be one from following:

```
1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL

5) SIGTRAP 6) SIGABRT 7) SIGBUS 8) SIGFPE

9) SIGKILL 10) SIGUSR1 11) SIGSEGV 12) SIGUSR2

13) SIGPIPE 14) SIGALRM 15) SIGTERM 17) SIGCHLD

18) SIGCONT 19) SIGSTOP 20) SIGTSTP 21) SIGTTIN

22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ

26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO

30) SIGPWR 31) SIGSYS 32) SIGRTMIN 33) SIGRTMIN+1

34) SIGRTMIN+2 35) SIGRTMIN+3 36) SIGRTMIN+4 37) SIGRTMIN+5

38) SIGRTMIN+6 39) SIGRTMIN+7 40) SIGRTMIN+8 41) SIGRTMIN+9

42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13

46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMAX-15 49) SIGRTMAX-14

50) SIGRTMAX-13 51) SIGRTMAX-12 52) SIGRTMAX-11 53) SIGRTMAX-10

54) SIGRTMAX-9 55) SIGRTMAX-8 56) SIGRTMAX-7 57) SIGRTMAX-6

58) SIGRTMAX-5 59) SIGRTMAX-4 60) SIGRTMAX-3 61) SIGRTMAX-2

62) SIGRTMAX-1 63) SIGRTMAX
```

In Windows, the only supported signal is SIGKILL with signal number 9.

relocator

Purpose:

The `relocator` interface allows to perform the following Virtuozzo Container migration operations:

- Physical to Virtual (P2V) migration.
- Virtual to Physical (V2P) migration.
- Virtual to Virtual (V2V) migration.

- In addition, the interface allows to clone Virtuozzo Containers, and to change the location of the Container data on the host machine.

This section describes the base `relocator` interface, which defines a generic migrator functionality. Virtuozzo Containers implementation of this interface is called `vzarelocator` (p. 604). When working with Virtuozzo Containers, P2V and V2P migration must be performed using `vzarelocator` (p. 604). V2V migration is the only migration operation that must be performed using the base `relocator` interface described in this section.

Types

clone_optionsType

Summary:

The `clone_optionsType` is the base type defining the options used during Virtuozzo Containers cloning operation.

Type specification:

The type has no elements.

Subtypes:

`vzarelocator.clone_optionsType` (p. 604)

p2v_migrate_optionsType

Summary:

Physical-to-virtual migration options.

Type specification:

Name	Min/Max	Type	Description
`exclude`	`0..1`		The list of files and directories on the source machine to exclude from migration. Physical disks to be excluded from migration.
`{`			
`path`	`1..[]`	`base64Binary`	Pathnames. The parameter uses the `rsync` exclude patterns. Disks letter including the colon, e.g. `E:`
`}`			

keep_dst	0..1	boolean	Specifies whether to keep the transferred data on the target host in case of error. Keeping the data may help reduce the duration of the subsequent migration operation. true -- keep the data. false -- remove the data.
config	0..1	env_configType	Configuration parameters to apply to the target Container. Normally, you should use the calc_env_config call (p. 393) to assist you in calculating the correct Container configuration based on the source physical machine parameters.
service	0..[]		The list of services to be stopped on the source machine during migration.
{			
name	1..1	string	Service name.
}			
quota	0..1		Disk quota migration information. You may specify only one partition on the source server which will be migrated to the Container together with all quotas imposed on it. All other partitions on the server will be copied without keeping their quota limits. Moreover, the quota limits imposed on the selected partition will be applied to the entire Container after the server migration.
{			
partition	1..1		Partition info.
{			
name	1..1	string	Partition name.
}			
}			

v2v_migrate_optionsType

Summary:

Virtual-to-virtual migration options.

Type specification:

Name	Min/Max	Type	Description
`force`	`0..1`	`none`	If the migration fails, you may try to force it by including this element. For example, if any of the application templates that are installed in the source Virtuozzo Container are missing on the destination Hardware Node, the migration will stop with an error. If you include the `force` option, the errors will be ignored and the Container will be migrated (note that it will be impossible to start such a Container after the migration).
`nostart`	`0..1`	`none`	A flag indicating not to start the target Container upon completion: `true` -- do not start the Container. `false` or absent -- start the Container.
`remove`	`0..1`	`boolean`	Indicates whether to delete all of the user data from the source Container. `true` -- delete the data. The Container private area will be deleted from the source node. `false` -- do not remove the data (default). The Container private area directory will be renamed and kept on the source node in its original location.
`config`	`0..1`	`env_configType`	New configuration parameters to apply to the target Container.
`type`	`0..1`	`int`	The type of migration to perform: `0` -- Offline `1` -- Simple online `2` -- Lazy online `3` -- Iterative online `4` -- Iterative lazy online

v2p_migrate_optionsType

Summary:

Virtual-to-physical migration options.

Type specification:

Name	Min/Max	Type	Description
`config`	`0..1`	`env_configType`	Configuration parameters to apply to the target physical server.

hw_notesType

Summary:

Miscellaneous information returned by the `calc_env_config` call (p. 393). See subtypes for plug-in specific implementations.

Type specification:

The type has no elements.

Subtypes:

`vzarelocator:hw_notesType` (p. 605)

Calls

Call	Description
migrate_p2v (p. 383)	Physical to virtual migration.
migrate_v2v (p. 389)	Virtual to virtual migration.
migrate_v2p (p. 392)	Virtual to physical migration.
calc_env_config (p. 393)	Calculates configuration parameters for P2V migration.
move (p. 397)	Moves a Container private area to a different location.
clone (p. 398)	Clones a Container.

migrate_p2v

Summary:

Migrates a physical server to a Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
migrate_p2v	1..1		
{			
options	0..1	p2v_migrate_optionsType (p. 379)	Migration options.
src	1..1	connection_infoType (p. 30)	The parameters that will be used to connect to the physical server that will be migrated to a Virtuozzo Container. The user authentication will be performed using the Agent authentication mechanism. The following parameters must be specified: protocol -- TCP or SSL. address -- the source node IP address. login/name -- the name of the system user, such as root for example. realm -- the ID of the System Realm. password -- the user password. port -- 4433 for TCP, or 4434 for SSL.
}			

Returns:

Name	Min/Max	Type	Description
eid	1..1	eid_type (p. 25)	Server ID of the new Virtuozzo Container.

Description:

The `migrate_p2v` call allows to migrate a standalone physical server running Linux, Windows Server 2003, or Windows 2000 system to a Virtuozzo Container. The migration process includes copying the entire contents of the physical server, including all files, folders, network settings, etc.

Virtuozzo Containers for Linux requirements and restrictions:

- The Linux distribution installed on the source physical server must be supported by Virtuozzo Containers (none of the BSD operating systems are supported).
- SSH must be installed on both the source server and the target Hardware Node. SSH is used to establish an initial connection with the source server and to copy the necessary Agent components to it.

For additional information, requirements, and limitations, please also see the Migrating Physical Server to Container section in the Virtuozzo Containers User's Guide.

Virtuozzo Containers for Windows requirements and restrictions:

1. You can only migrate physical servers running Windows Server 2003 or Windows 2000 systems.
2. If you are migrating a server running Windows Server 2003, you should ascertain that:
   - The target Hardware Node is running the same version and Service Pack of Windows Server 2003 as the source machine.
   - The language version of Windows Server 2003 is the same on both machines.
3. If you are migrating a physical server running Windows 2000, please keep the following facts in mind:
   - The Windows components that are installed on the source server will not be migrated if they are not included in the Windows Server 2003 OS template on the target Hardware Node.
   - Some applications may fail to work properly on the target Container because of the differences in the Windows Server 2003 technology in comparison to Windows 2000.

4 You cannot migrate physical servers running the 32-bit version of Windows Server 2003/Windows Server 2000 to Hardware Nodes running the 64-bit version of Virtuozzo Containers.

5 You cannot migrate physical servers running the 64-bit version of Windows Server 2003/Windows Server 2000 to Nodes running the 32-bit version of Virtuozzo Containers.

6 You cannot migrate non-NTFS partitions.

For additional information, requirements, and limitations, please also see the Migrating Physical Server to Container section in the Virtuozzo Containers for Windows User's Guide.

Prior to executing the `migrate_p2v` call, you have to take the following preliminary steps:

1 Distribute the core Agent components to the source server. Use the `system/distribute` (p. 530) call to accomplish this task.

2 Calculate the Container configuration parameters using the `calc_env_config` call (p. 393). Please note that the call returns the configuration information using the `env_configType` data type but you must use its subtype, the `venv_configType` (p. 61) type to pass this data to the `migrate_p2v` call (p. 383) (see code examples below).

Not all of the configuration parameters returned by the `calc_env_config` call (p. 393) can be effectively used to create the new Container. Most notably, the IP address contained in the `address` element of the returned XML packet is a read-only field and has no effect when creating a new Conatiner. If you would like to specify an IP address for the new Container, use the `net_device` element of the `venv_configType` structure (p. 61) instead. You might also want to specify the Virtuozzo-level Container ID and an OS template values. Once you are done editing the Container configuration, include it in the request using the `options/config` parameter (see the Request Specification subsection above).

On successful migration, the source physical server will be automatically stopped to avoid possible conflicts with the new Container.

Example:

1. The first step is to distribute the core Agent components to the source server (see `distribute` (p. 530) for code example).

2. The second step is to calculate the Container configuration based on the source server (see `calc_env_config` (p. 393) for code example). In the example below, we've modified the calculated configuration by adding the `veid`, `os_template`, and `net_device` parameters containing the Virtuozzo-level Container ID, OS template name, and network interface/IP address information.

Now that all of the necessary preliminary steps are completed, we are ready to execute the `migrate_p2v` call.

Input

```
<packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzarelocator"
xmlns:ns4="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzarelocator</target>
  <data>
    <vzarelocator>
      <migrate_p2v>
        <options>
          <exclude>
            <path>/vz</path>
          </exclude>
          <keep_dst/>
          <ns2:config xsi:type="ns4:venv_configType">
            <ns3:qos>
              <ns3:id>avnumproc</ns3:id>
              <ns3:hard>94</ns3:hard>
              <ns3:soft>94</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>cpuunits</ns3:id>
              <ns3:hard>1000</ns3:hard>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>dcachesize</ns3:id>
              <ns3:hard>1270579</ns3:hard>
              <ns3:soft>1270579</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>dgramrcvbuf</ns3:id>
              <ns3:hard>132096</ns3:hard>
              <ns3:soft>132096</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>diskinodes</ns3:id>
              <ns3:hard>69369</ns3:hard>
              <ns3:soft>63063</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>diskspace</ns3:id>
              <ns3:hard>2156412</ns3:hard>
              <ns3:soft>1960375</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>kmemsize</ns3:id>
              <ns3:hard>7750533</ns3:hard>
              <ns3:soft>7750533</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>lockedpages</ns3:id>
              <ns3:hard>1261</ns3:hard>
              <ns3:soft>1261</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numfile</ns3:id>
              <ns3:hard>3008</ns3:hard>
              <ns3:soft>3008</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numflock</ns3:id>
              <ns3:hard>110</ns3:hard>
              <ns3:soft>100</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numiptent</ns3:id>
              <ns3:hard>128</ns3:hard>
              <ns3:soft>128</ns3:soft>
```

```
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numothersock</ns3:id>
              <ns3:hard>94</ns3:hard>
              <ns3:soft>94</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numproc</ns3:id>
              <ns3:hard>94</ns3:hard>
              <ns3:soft>94</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numpty</ns3:id>
              <ns3:hard>255</ns3:hard>
              <ns3:soft>255</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numsiginfo</ns3:id>
              <ns3:hard>256</ns3:hard>
              <ns3:soft>256</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>numtcpsock</ns3:id>
              <ns3:hard>80</ns3:hard>
              <ns3:soft>80</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>oomguarpages</ns3:id>
              <ns3:hard>2147483647</ns3:hard>
              <ns3:soft>0</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>othersockbuf</ns3:id>
              <ns3:hard>372736</ns3:hard>
              <ns3:soft>132096</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>physpages</ns3:id>
              <ns3:hard>2147483647</ns3:hard>
              <ns3:soft>0</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>privvmpages</ns3:id>
              <ns3:hard>27074</ns3:hard>
              <ns3:soft>22562</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>quotatime</ns3:id>
              <ns3:hard>0</ns3:hard>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>quotaugidlimit</ns3:id>
              <ns3:hard>1024</ns3:hard>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>shmpages</ns3:id>
              <ns3:hard>8192</ns3:hard>
              <ns3:soft>8192</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>tcprcvbuf</ns3:id>
              <ns3:hard>270336</ns3:hard>
              <ns3:soft>65536</ns3:soft>
            </ns3:qos>
            <ns3:qos>
              <ns3:id>tcpsndbuf</ns3:id>
              <ns3:hard>270336</ns3:hard>
```

```
            <ns3:soft>65536</ns3:soft>
          </ns3:qos>
          <ns3:qos>
            <ns3:id>vmguarpages</ns3:id>
            <ns3:hard>2147483647</ns3:hard>
            <ns3:soft>0</ns3:soft>
          </ns3:qos>
          <ns3:distribution>
            <ns3:version>5</ns3:version>
            <ns3:name>rhel</ns3:name>
          </ns3:distribution>
          <ns3:hostname>dhcp0-36.sw.ru</ns3:hostname>
          <ns3:search_domain>sw.ru</ns3:search_domain>
          <ns3:nameserver>192.168.1.1</ns3:nameserver>
          <veid>201</veid>
          <os_template>
            <name>redhat-as3-minimal</name>
          </os_template>
          <net_device>
            <id>venet0</id>
            <ip_address>
              <ip>10.17.3.123</ip>
            </ip_address>
            <host_routed/>
          </net_device>
        </ns2:config>
      </options>
      <src>
        <protocol>SSL</protocol>
        <address>192.168.0.36</address>
        <login>
          <name>cm9vdA==</name>
          <realm>00000000-0000-0000-0000-000000000000</realm>
        </login>
        <password>bXlwYXNz</password>
        <port>4434</port>
      </src>
    </migrate_p2v>
  </vzarelocator>
 </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzarelocator"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="2ac465c3671t1547rc38" time="2007-05-25T09:04:46+0000" priority="4000"
version="4.0.0">
<ns1:origin>vzarelocator</ns1:origin>
<ns1:target>vzclient25-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzarelocator>
    <ns2:eid>0cb73fc2-9925-0841-809a-8bf04794856d</ns2:eid>
  </ns2:vzarelocator>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

migrate_v2v

Summary:

Migrates a Virtuozzo Container from one Hardware Node to another.

Request specification:

Name	Min/Max	Type	Description
`migrate_v2v`	`1..1`		
`{`			
`options`	`0..1`	`v2v_migrate_optionsType (p. 381)`	Migration options.
`eid`	`1..1`	`eid_type (p. 25)`	Server ID of the source Container.
`dst`	`1..1`	`connection_infoType (p. 30)`	The destination Hardware Node connection information. You must specify the `name`, `realm`, and `password` parameters that are valid on the destination Hardware Node. If, for example, you are authenticating the user on the target node against a certain Realm but don't know the Realm ID in advance, you must connect to the destination node first and get the Realm ID using the `get_realm` call (p. 99). Alternately, you may use the `generate_pass` call (p. 532) to get a temporary login info and use the returned values to populate this structure. The `generate_pass` call must also be executed on the target Hardware Node.
`}`			

Returns:

OK/Error

Description:

To migrate a Virtuozzo Container, the target host must have Virtuozzo Containers and Parallels Agent software installed.

The following V2V migration types are supported:

- *Offline migration*. Performed on a stopped or a running source Container. If the Container is stopped, all its files are simply copied from the source host to the target computer. If the Container is running, the files are first copied to the target machine and then the Container is stopped momentarily. At this point, the data that was copied to the target machine is compared to the original data and the files that have changed since the copying began are updated. The source Container is then started back up. The downtime depends on the Container size but should normally take only a minute or so.
- *Simple online migration*. Performed on a running source Container. In the beginning of the migration process, the Container becomes momentarily locked and all of its data, including states of all running processes, is dumped into an image file. After that, the Container operation is resumed, and the dump file is transferred to the target host where a new Container is created using the file data.
- *Lazy online migration*. Instead of migrating all of the data in one big step (as in simple online migration), lazy migration copies the data over a time period. Initially, only the data that is absolutely necessary to bring the new Container up is copied to the target host. The rest of the data remains locked on the source host and is copied to the destination host on as-needed basis. By using this approach, you can decrease the services downtime to near zero.
- *Iterative online*. During the iterative online migration, the Container memory is transferred to the destination node before the Container data is dumped into an image file. Using this type of online migration allows to attain the smallest service delay.
- *Iterative + lazy online migration*. This type of online migration combines the techniques used in both lazy and iterative migration types, i.e. some part of the Container memory is transferred to the destination host before dumping a Container, and the rest of the data is transferred on-demand after the Container has been successfully created on the target host.

On successful migration, the original Container will no longer exist on the source node. This is done in order to avoid possible conflicts that may occur if both the original and the new Containers are running at the same time. However, the original Container data will NOT be deleted from the source Hardware Node. By default, the data is kept in its original location (the Container private area) but the private area directory itself will be renamed. You can completely remove the original Container data from the source node by including the `options/remove` parameter in the request.

Example:

Performing an offline migration (type 0). The `progress` argument is set to "on" to receive the migration progress messages.

Input

```
<packet version="4.0.0" progress="on">
  <target>relocator</target>
  <data>
    <relocator>
      <migrate_v2v>
        <options>
          <force/>
          <nostart/>
          <type>0</type>
        </options>
        <eid>a8a73d01-2969-6b48-9544-5cbf6ea7c554</eid>
        <dst>
          <protocol>SSL</protocol>
          <address>192.168.0.240</address>
          <login>
            <name>cm9vdA==</name>
            <realm>00000000-0000-0000-0000-000000000000</realm>
          </login>
          <password>bXlwYXNz</password>
          <port>4434</port>
        </dst>
      </migrate_v2v>
    </relocator>
  </data>
</packet>
```

Output

The following is an example of a progress message received during the migration process:

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="4fc46559f47tbdbrd64"
time="2007-05-22T02:02:58+0000" type="1" priority="4000" version="4.0.0">
<ns1:origin>relocator</ns1:origin>
<ns1:target>vzclient7-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:progress>
    <ns1:message>
      <ns1:message>Q2FsbGluZyB2em1zcmMuLi4=</ns1:message>
      <ns1:name></ns1:name>
      <ns1:translate/>
      <ns1:parameter>
        <ns1:message>JXRpdGxlJQ==</ns1:message>
        <ns1:name>source_env</ns1:name>
        <ns1:translate/>
        <ns1:parameter>

<ns1:message>MWRmNGIwNGUtMGQ1NS1mMjQ2LWI3MTgtODliYmM2MmZkMzcx</ns1:message>
          <ns1:name>eid</ns1:name>
          <ns1:translate/>
        </ns1:parameter>
        <ns1:parameter>
          <ns1:message>ZGhjcDAtMTM2LnN3LnJ1</ns1:message>
          <ns1:name>title</ns1:name>
          <ns1:translate/>
        </ns1:parameter>
        <ns1:parameter>
          <ns1:message>Z2VuZXJpYw==</ns1:message>
          <ns1:name>type</ns1:name>
```

```
          <ns1:translate/>
        </ns1:parameter>
      </ns1:parameter>
    </ns1:message>
    <ns1:status>2</ns1:status>
  </ns1:progress>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

On successful migration, you will receive the standard Agent "OK" message.

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/relocator"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="4fc46559f47tbdbrd64"
time="2007-05-22T02:03:25+0000" priority="4000" version="4.0.0">
<ns1:origin>relocator</ns1:origin>
<ns1:target>vzclient7-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:relocator>
    <ns1:ok/>
  </ns2:relocator>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

migrate_v2p

Summary:

Migrates a Virtuozzo Container to a physical server.

Request specification:

Name	Min/Max	Type	Description
migrate_v2p	1..1		
{			
options	0..1	v2p_migrate_optionsType (p. 382)	Migration options.
eid	1..1	eid_type (p. 25)	The Server ID of the source Virtuozzo Container.
dst	1..1	connection_infoType (p. 30)	Destination physical server connection information.
}			

Returns:

OK/Error

Example:

```
<packet version="4.0.0">
  <target>vzarelocator</target>
  <data>
    <vzarelocator>
      <migrate_v2p>
        <eid>9bafbeb7-85f7-499e-a210-40e00850a5f3</eid>
        <dst>
          <protocol>TCP</protocol>
          <address>192.168.0.64</address>
          <login>
            <name>root</name>
            <realm>00000000-0000-0000-0000-000000000000</realm>
          </login>
          <password>bXlwYXNz</password>
        </dst>
      </migrate_v2v>
    </vzarelocator>
  </data>
</packet>
```

calc_env_config

Summary:

Calculates Virtuozzo Container configuration parameters for P2V migration.

Request specification:

Name	Min/Max	Type	Description
calc_env_config	1..1	none	

Returns:

Name	Min/Max	Type	Description
config	1..1	env_configType	Calculated configuration data.
hw_notes	1..1	hw_notesType (p. 382)	Miscellaneous information.

Description:

The `calc_env_config` call is used prior to migrating a physical server to a Virtuozzo Container. The call calculates the target Container configuration based on the source physical server parameters. The resulting configuration is then passed to the `migrate_p2v` call (p. 383).

The call must be processed on the source physical server that you want to migrate to a Virtuozzo Container. In order to do that, you first have to distribute the necessary Agent components to the physical server using the `system/distribute` call (p. 530). After that, connect to the Agent that you just distributed and execute the `calc_env_config` call to calculate the Container configuration. Please note that the call works only on non-Virtuozzo servers on which Agent core was installed using the `system/distribute` call (p. 530). It will not work on servers hosting Virtuozzo Containers and running a full version of Parallels Agent software. **Example:**

Input

```
<packet version="4.0.0">
  <target>vzarelocator</target>
  <data>
    <vzarelocator>
      <calc_env_config/>
    </vzarelocator>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns4="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzarelocator"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="bc4677c76at3d6crc2c"
time="2007-06-19T15:38:15+0000" priority="0" version="4.0.0">
<ns1:origin>vzarelocator</ns1:origin>
<ns1:target>vzclient3-c613d7dc-d122-f04f-98f5-815aea54456f</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzarelocator>
    <ns2:config xsi:type="ns3:venv_configType">
      <ns4:qos>
        <ns4:id>avnumproc</ns4:id>
        <ns4:hard>93</ns4:hard>
        <ns4:soft>93</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>cpuunits</ns4:id>
        <ns4:hard>1000</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>dcachesize</ns4:id>
        <ns4:hard>1257062</ns4:hard>
        <ns4:soft>1257062</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>dgramrcvbuf</ns4:id>
        <ns4:hard>132096</ns4:hard>
        <ns4:soft>132096</ns4:soft>
      </ns4:qos>
```

```
      <ns4:qos>
        <ns4:id>diskinodes</ns4:id>
        <ns4:hard>69402</ns4:hard>
        <ns4:soft>63093</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>diskspace</ns4:id>
        <ns4:hard>2250937</ns4:hard>
        <ns4:soft>2046307</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>kmemsize</ns4:id>
        <ns4:hard>7668080</ns4:hard>
        <ns4:soft>7668080</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>lockedpages</ns4:id>
        <ns4:hard>1260</ns4:hard>
        <ns4:soft>1260</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numfile</ns4:id>
        <ns4:hard>2976</ns4:hard>
        <ns4:soft>2976</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numflock</ns4:id>
        <ns4:hard>110</ns4:hard>
        <ns4:soft>100</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numiptent</ns4:id>
        <ns4:hard>128</ns4:hard>
        <ns4:soft>128</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numothersock</ns4:id>
        <ns4:hard>93</ns4:hard>
        <ns4:soft>93</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numproc</ns4:id>
        <ns4:hard>93</ns4:hard>
        <ns4:soft>93</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numpty</ns4:id>
        <ns4:hard>255</ns4:hard>
        <ns4:soft>255</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numsiginfo</ns4:id>
        <ns4:hard>256</ns4:hard>
        <ns4:soft>256</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>numtcpsock</ns4:id>
        <ns4:hard>80</ns4:hard>
        <ns4:soft>80</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>oomguarpages</ns4:id>
        <ns4:hard>2147483647</ns4:hard>
        <ns4:soft>0</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>othersockbuf</ns4:id>
```

```
        <ns4:hard>370176</ns4:hard>
        <ns4:soft>132096</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>physpages</ns4:id>
        <ns4:hard>2147483647</ns4:hard>
        <ns4:soft>0</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>privvmpages</ns4:id>
        <ns4:hard>27033</ns4:hard>
        <ns4:soft>22528</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>quotatime</ns4:id>
        <ns4:hard>0</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>quotaugidlimit</ns4:id>
        <ns4:hard>1024</ns4:hard>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>shmpages</ns4:id>
        <ns4:hard>8192</ns4:hard>
        <ns4:soft>8192</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>tcprcvbuf</ns4:id>
        <ns4:hard>270336</ns4:hard>
        <ns4:soft>65536</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>tcpsndbuf</ns4:id>
        <ns4:hard>270336</ns4:hard>
        <ns4:soft>65536</ns4:soft>
      </ns4:qos>
      <ns4:qos>
        <ns4:id>vmguarpages</ns4:id>
        <ns4:hard>2147483647</ns4:hard>
        <ns4:soft>0</ns4:soft>
      </ns4:qos>
      <ns3:distribution>
        <ns3:version>5</ns3:version>
        <ns3:name>rhel</ns3:name>
      </ns3:distribution>
      <ns4:address>
        <ns4:ip>192.168.0.188</ns4:ip>
      </ns4:address>
      <ns4:hostname>dhcp0-188.sw.ru</ns4:hostname>
      <ns4:search_domain>sw.ru</ns4:search_domain>
      <ns4:nameserver>192.168.1.1</ns4:nameserver>
    </ns2:config>
    <ns2:hw_notes xsi:type="ns2:hw_notesType">
      <ns2:exclude>
        <ns2:path>L2Rldi9wdHMvKg==</ns2:path>
        <ns2:discardable>0</ns2:discardable>
      </ns2:exclude>
      <ns2:exclude>
        <ns2:path>L2Rldi9zaG0vKg==</ns2:path>
        <ns2:discardable>0</ns2:discardable>
      </ns2:exclude>
      <ns2:exclude>
        <ns2:path>L3Byb2MvKg==</ns2:path>
        <ns2:discardable>0</ns2:discardable>
      </ns2:exclude>
      <ns2:exclude>
        <ns2:path>L3N5cy8q</ns2:path>
```

```
        <ns2:discardable>0</ns2:discardable>
      </ns2:exclude>
      <ns2:exclude>
        <ns2:path>L3Zhci9saWIvbmZzL3JwY19waXBlZnMvKg==</ns2:path>
        <ns2:discardable>0</ns2:discardable>
      </ns2:exclude>
    </ns2:hw_notes>
  </ns2:vzarelocator>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

move

Summary:

Allows to change a Virtuozzo-level Container ID and the location of the Container data on the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
move	1..1		
{			
eid	1..1	eid_type (p. 25)	Server ID of the source Virtuozzo Container.
config	1..1	env_configType	Use this element to set the new Virtuozzo-level Container ID and/or the new root and private area directories. You cannot modify any other parameters provided in the config structure. The move call allows to modify just the Container ID (veid) and the data locations. All other configuration parameters will be ignored.
}			

Returns:

OK/Error

Description:

This call provides a simple way of changing the Container ID, the locations of the Container private area directory, and the Container root directory. The pathnames of the private area and root directories by default include the Container ID, e.g. `/vz/root/205` (205 here is the Virtuozzo-level Container ID). Although the actual names of these directories do not directly affect the Container operations, you should still change them (for consistency reasons) if you are changing the ID of a Virtuozzo Container. Please note that the Server ID (globally unique ID) of the Container will also change after any of the modifications performed by this call.

Example:

```
<packet xmlns:ns4="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzarelocator</target>
  <data>
    <vzarelocator>
      <move>
        <eid>a8a73d01-2969-6b48-9544-5cbf6ea7c554</eid>
        <config xsi:type="ns4:venv_configType">
          <veid>205</veid>
          <ve_root>/vz/root/205</ve_root>
          <ve_private>/vz/private/205</ve_private>
        </config>
      </move>
    </vzarelocator>
  </data>
</packet>
```

clone

Summary:

Creates the specified number of exact copies of the specified Virtuozzo Container on the same Hardware Node.

Request specification:

Name	Min/Max	Type	Description
clone	1..1		
{			
eid	1..1	eid_type (p. 25)	The ID of the Container to clone.
count	1..1	int	The number of copies to create.
options	0..1	clone_optionsType (p. 379)	Cloning options.
}			

Returns:

Name	Min/Max	Type	Description
`eid_list`	`1..1`	`eid_listType (p. 32)`	A list containing the Server IDs of the created Containers.

Description:

In general, you can clone a running or a stopped Container. If, for any reason, a running Container cannot be cloned, you will receive an error message advising you to stop the Container first.

Example:

The following example creates three copies of the specified Container.

Input

```
<packet xmlns:ns4="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:ns5="http://www.swsoft.com/webservices/vza/4.0.0/vzarelocator"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzarelocator</target>
  <data>
    <vzarelocator>
      <clone>
        <eid>642e422d-94e8-5a4b-8a58-805f539ffb6e</eid>
        <count>3</count>
        <options xsi:type="ns5:clone_optionsType">
          <config xsi:type="ns4:venv_configType">
            <ve_root>/vz/root/$VEID</ve_root>
            <ve_private>/vz/private/$VEID</ve_private>
          </config>
          <veid>500</veid>
          <veid>501</veid>
          <veid>502</veid>
        </options>
      </clone>
    </vzarelocator>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzarelocator"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="6cc4656b607t5f49rd64" time="2007-05-22T15:00:31+0000" priority="4000"
version="4.0.0">
<ns1:origin>vzarelocator</ns1:origin>
<ns1:target>vzclient17-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzarelocator>
    <ns2:eid_list>
      <ns2:eid>c92e5d4d-6c2f-6746-a8f5-802e954f8412</ns2:eid>
      <ns2:eid>466cf488-0f95-8240-be7a-d005bcbc47ed</ns2:eid>
      <ns2:eid>14b9b439-16f6-eb46-9d19-4b8939a22f78</ns2:eid>
    </ns2:eid_list>
  </ns2:vzarelocator>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

res_log

Purpose:

The system resources log management interface. The utilization server resources, such as CPU, memory, network, etc., is automatically logged in the history database. The `res_log` interface provides calls that allow to access this information.

Types

classType

Summary:

A performance class, instance, and counter information. See `perf_mon:classType` (p. 353) for more info on performance classes.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	Performance class name.
`instance`	`0..1`	`string`	Class instance.
`counter`	`0..1`	`string`	Performance counter.

logType

Summary:

Indicates the logging period for a performance class.

Type specification:

Name	Min/Max	Type	Description
`class`	`1..[]`	restriction: `classType` (p. 401)	Performance class.
`{`			
`name`	`1..1`	`string`	Class name.
`}`			
`period`	`1..1`	`int`	Logging period in seconds.

Calls

Call	Description
get_log (p. 402)	Retrieves performance logs for the specified server.
set_log (p. 405)	Sets logging period for the specified classes.
get_log_info (p. 407)	Provides information about current logging.
get_top (p. 409)	Returns information about the top resource-consuming Environments.

get_log

Summary:

Retrieves performance logs for the specified server.

Request specification:

Name	Min/Max	Type	Description
`get_log`			
{			
`eid`	`1..1`	`eid_type` (p. 25)	The ID of the server to get the data for.
`class`	`0..[]`	`classType` (p. 401)	Performance class. If this element is omitted, retrieves logged data for all classes. The classes specified here must be compatible with the type of the server specified in the `eid` element. See `perf_mon:start_monitor` (p. 355) for more info on classes.
`start_time`	`0..1`	`datetime_type`	Start time of the log.
`end_time`	`0..1`	`datetime_type`	End time of the log.
[			*The following two elements are not the children of the end_time element. They are a separate choice group.*
`records`	`0..1`	`int`	Number of records to retrieve beginning from the most recent record going back through time. The records will appear in the reverse chronological order, i.e. the most recent record will appear first in the list.

period	0..1	int	Period of logging, i.e. time in seconds between the two neighbouring results in the log. Most of the time the recorded periods in the database will not be equal to the requested ones, so recalculation and approximation will be used. If an interval doesn't have a value, it means that there is no information in the log from which the result can be approximated.
]			
report_empty	0..1	none	If this element is included then absent data intervals will be reported explicitly.
}			

Returns:

Name	Min/Max	Type	Description
data	0..[]	perf_dataType (p. 51)	Performance data.

Example:

Retrieving performance logs for the specified server for the class `counters_vz_cpu`.

Input

```
<packet version="4.0.0">
  <target>res_log</target>
  <data>
    <res_log>
      <get_log>
        <eid>565b96bd-d2da-4c7e-a212-0943a4bd6b29</eid>
        <class>
          <name>counters_vz_cpu</name>
        </class>
        <records>5</records>
        <report_empty/>
      </get_log>
    </res_log>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="3ec4683b8e5t1547rb2c" time="2007-06-28T17:22:55+0000" priority="0"
version="4.0.0">
<ns1:origin>perf_mon</ns1:origin>
<ns1:target>vzclient1-6d9ea6b6-e470-424b-98ca-27dd10e49860</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:perf_mon>
    <ns1:data>
      <ns1:eid>1649d32f-3e18-6642-9690-4fe3cd406eb0</ns1:eid>
      <ns1:interval>
        <ns1:start_time>2007-06-28T17:22:35+0000</ns1:start_time>
        <ns1:end_time>2007-06-28T17:22:55+0000</ns1:end_time>
      </ns1:interval>
      <ns1:class>
        <ns1:name>counters_vz_cpu</ns1:name>
        <ns1:instance>
          <ns1:name></ns1:name>
          <ns1:counter>
            <ns1:name>counter_cpu_system</ns1:name>
            <ns1:value>
              <ns1:avg>180</ns1:avg>
              <ns1:min>87</ns1:min>
              <ns1:max>87</ns1:max>
              <ns1:cur>87</ns1:cur>
            </ns1:value>
          </ns1:counter>
        </ns1:instance>
      </ns1:class>
    </ns1:data>
  </ns1:perf_mon>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

set_log

Summary:

Sets the logging period for the specified performance class.

Request specification:

Name	Min/Max	Type	Description
`set_log`	`1..1`	`logType (p. 401)`	The class and the logging period information.

Returns:

OK/Error

Description:

System performance data is automatically collected by periodic collector operators at predefined time intervals. The default interval is 60 minutes. The `set_log` call allows you to modify the interval for each performance class individually. When you execute the call on the node that hosts Virtuozzo Containers, the change affects the host itself and all of the Containers hosted by it. See `perf_mon:start_monitor` (p. 355) for more info on performance classes. To get the current logging period settings, use the `get_log_info` call (p. 407).

Example:

Setting the logging period for the `counters_cpu` class to 1000 seconds.

Input

```
<packet version="4.0.0">
  <target>res_log</target>
  <data>
    <res_log>
      <set_log>
        <class>
          <name>counters_cpu</name>
        </class>
        <period>1000</period>
      </set_log>
    </res_log>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/res_log"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="7c45f6c95at72aeraa4"
time="2007-03-11T08:54:59+0000" priority="0" version="4.0.0">
<ns1:origin>res_log</ns1:origin>
<ns1:target>vzclient7</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:res_log>
    <ns1:ok/>
  </ns2:res_log>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_log_info

Summary:

Retrieves a list of performance classes with logging periods.

Request specification:

Name	Min/Max	Type	Description
get_log_info	1..1	none	

Returns:

Name	Min/Max	Type	Description
log_info	0..[]	logType (p. 401)	Current logging information.

Description:

System performance data is automatically collected by periodic collector operators at predefined time intervals. The default interval is 60 minutes but can be modified using the `set_log` call (p. 405). The `get_log_info` call allows you retrieve the current logging period settings. See `perf_mon:start_monitor` (p. 355) for more info on performance classes.

Example:

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>res_log</target>
  <data>
    <res_log>
      <get_log_info/>
    </res_log>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/res_log"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="56c45f6a71ft1366rdfc" time="2007-03-11T07:21:54+0000" priority="0"
version="4.0.0">
<ns1:origin>res_log</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:res_log>
    <ns2:log_info>
      <ns2:class>counters_cpu</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_disk</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_loadavg</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_memory</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_net</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_process</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_system</ns2:class>
      <ns2:period>3600</ns2:period>
```

```
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_cpu</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_hw_net</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_loadavg</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_memory</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_net</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_process</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_quota</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_slm</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_system</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
    <ns2:log_info>
      <ns2:class>counters_vz_ubc</ns2:class>
      <ns2:period>3600</ns2:period>
    </ns2:log_info>
  </ns2:res_log>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_top

Summary:

Retrieves a list of the top resource consuming servers.

Request specification:

Name	Min/Max	Type	Description
get_top	1..1		
{			

parameter	1..1	string	The name of the "top" counter to use. The counter must be compatible with the servers specified in the eid_list element below.
count	0..1	int	The maximum number of servers to include in the report.
descending	0..1	none	Include this element to sort the result set by Server ID (eid) in descending order.
start_time	0..1	datetime_type	Start time of the log.
end_time	0..1	datetime_type	End time of the log.
eid_list	0..1	eid_listType (p. 32)	The IDs of the servers to include in the report. If this element is omitted, all known servers will be included (the total number of the servers may be limited by the value provided in the count element if present).
exclude	0..1	none	If this element is included, the list specified in the eid_list element will be treated as the list of the servers to exclude from the result.
}			

Returns:

Name	Min/Max	Type	Description
top	0..1		A list of the top resource-consuming servers.
{			
set	0..[]		
{			
eid	1..1	eid_type (p. 25)	Server ID.
value	1..1	anySimpleType	Resource consumption value.
}			
total	1..1	anySimpleType	An aggregate resource consumption value for all servers.
}			

Description:

The resource type for which you want to retrieve the data is specified in the `parameter` element and is the name of one of the "top" counters. The following table lists the available counters.

Virtuozzo Containers-specific counters

Counter name	Type	Units	Description
CPU			
`counter_cpu_used`	`float`	percent	Total CPU usage.
`counter_cpu_share_used`	`float`	percent	The real CPU usage of a Container against the CPU limit set for it.
Disk			
`counter_disk_used`	`int`	bytes	The amount of disk space in use.
`counter_disk_share_used`	`float`	percent	The ratio of the real disk space consumption by a Container against the disk space limit set for it.
Memory			
`counter_memory_used`	`int`	bytes	The total amount of memory used by a Container.
`counter_memory_share_used`	`float`	percent	The ratio of the real physical memory usage of a Container against the memory limit set for it.

Generic counters

Counter name	Type	Units	Description
CPU			

counter_cpu_used	float	percent	Total CPU usage.
counter_cpu_share_used	float	percent	The ratio of CPU time consumed by a server to current limit.
Disk			
counter_disk_used	int	bytes	The amount of disk space in use.
counter_disk_share_used	float	percent	The ratio of used disk space to current limit.
Memory			
counter_memory_used	int	bytes	Memory used by a server.
counter_memory_share_used	float	percent	The ratio of the real physical memory usage of a server against the memory limit set for it.
Network			
counter_net_incoming_bytes	int	bytes	Amount of incoming network traffic in bytes.
counter_net_incoming_packets	int	pcs	Amount of incoming network traffic in packets.
counter_net_outgoing_bytes	int	bytes	Amount of outgoing network traffic in bytes.
counter_net_outgoing_packets	int	pcs	Amount of outgoing network traffic in packets

Example:

The following example shows how to retrieve the top CPU consuming servers.

Input

```
<packet version="4.0.0">
  <target>res_log</target>
  <data>
    <res_log>
      <get_top>
         <parameter>counter_cpu_used</parameter>
          <descending/>
      </get_top>
    </res_log>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/res_log"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0"
priority="0" id="6c48773cb6t2cd6r175c" time="2008-07-11T14:02:22+0000">
<origin>res_log</origin>
<target>vzclient8-67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <res_log>
    <top>
      <set>
        <eid>67cefb4a-9a6e-2d41-94aa-cbfd20fa3943</eid>
        <value>1.225704</value>
      </set>
      <set>
        <eid>a818c39a-05cc-6446-b7e5-110a795b38d2</eid>
        <value>0.352999</value>
      </set>
      <set>
        <eid>d69e58c3-ad4a-8841-8115-7e0388b4e52d</eid>
        <value>0.018739</value>
      </set>
      <set>
        <eid>44db5492-7942-f743-8cdf-4435928f309c</eid>
        <value>0.003821</value>
      </set>
      <set>
        <eid>856bb909-85b6-104b-ac59-32e3160c2bde</eid>
        <value>0.000492</value>
      </set>
      <set>
        <eid>ef3c252a-b342-3448-b4dc-591bb7ec6082</eid>
        <value>0.000000</value>
      </set>
      <total>1.601756</total>
    </top>
  </res_log>
</data>
<src>
  <director>gend</director>
</src>
</packet>
```

resourcem

Purpose:

The `resourcem` interface gives the ability to manage pools of resources in a Virtuozzo group.

Types

resource_poolType

Summary:

The base resource type structure. The subtypes of this type extend it adding elements that are specific to their respective resource types.

Type specification:

Name	Min/Max	Type	Description
`type`	`1..1`	`string`	Resource type. See the subtypes of `resource_poolType` for more information.

Subtypes:

`resource_ip_poolType` (p. 415)

resourceType

Summary:

Generic resource type.

Type specification:

The type has no elements.

Subtypes:

`resource_ipType` (p. 416)

resource_ip_poolType

Summary:

Contains the IP address pool configuration information.

Type specification:

Extends `resource_poolType` (p. 414).

Adds the following elements:

Name	Min/Max	Type	Description
`type`	`1..1`	`string`	Resource type. The type of the IP resource is `resource_ip`.
`[`	`0..1`		Denotes a choice between `ip_range` and `ip`. You can specify a single IP address or a range of IP addresses.
`ip_range`	`1..1`		A range of IP addresses.
`{`			
`start_ip`	`1..1`	`ip_type` (p. 26)	First IP address in the range.
`end_ip`	`1..1`	`ip_type` (p. 26)	Last IP address in the range.
`}`			
`ip`	`1..1`	`ip_type` (p. 26)	A single IP address.
`]`			

Example:

The IP address range.

```
<type>resource_ip</type>
<ip_range>
  <start_ip>10.17.3.121</start_ip>
  <end_ip>10.17.3.125</end_ip>
</ip_range>
```

A single IP address

```
<type>resource_ip</type>
<ip>
  <start_ip>10.17.3.121</start_ip>
</ip>
```

resource_ipType

Summary:

Describes the IP address resource.

Type specification:

Extends `resourceType (p. 414)`

Adds the following elements:

Name	Min/Max	Type	Description
`ip`	`1..1`	`ip_type (p. 26)`	IP address.

Calls

Call	Description
add_resource (p. 417)	Add resource configuration to the specified pool(s).
remove_resource (p. 418)	Remove resource definition from the specified pool.
set_pool (p. 419)	Replace resource pool configuration.
get_pool (p. 420)	Get configuration for the specified pool(s).
allocate (p. 422)	Allocates a resource from the specified pool.
release (p. 425)	Release an allocated resource.

add_resource

Summary:

Adds a resource to a resource pool.

Request specification:

IP address pool

Name	Min/Max	Type	Description
`add_resource`	`1..1`		
`{`			
`resource_pool`	`1..[]`	`resource_poolType` (p. 414)	The resource configuration. Use the appropriate subtype of `resource_poolType` (p. 414) to add a resource to the desired pool.
`}`			

Returns:

OK/Error

Description:

The call adds a resource to a resource pool. It does not check the validity of the resource or its general availability (other than basic syntax and format checking). It does check if the resource being added already exists in the pool.

Example:

Adding a range of IP address to the IP address pool.

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <add_resource>
        <resource_pool xsi:type="ns2:resource_ip_poolType">
          <type>resource_ip</type>
   k       <ip_range>
            <start_ip>10.17.3.125</start_ip>
            <end_ip>10.17.3.127</end_ip>
          </ip_range>
        </resource_pool>
      </add_resource>
    </resourcem>
  </data>
</packet>
```

remove_resource

Summary:

Removes a resource from resource pool.

Request specification:

Name	Min/Max	Type	Description
`remove_resource`	`1..1`		
`{`			
`resource_pool`	`1..[]`	`resource_poolType` (p. 414)	Resource to remove from a pool. Use the appropriate subtype of `resource_poolType` (p. 414) to remove a resource from the desired pool.
`}`			

Returns:

OK/Error

Description:

The call removes a resource from a pool. It does not check if the resource is in use. You can remove a single value or a range of resource values.

Example:

Removing a single IP address from the range that was previously added using the `add_resource` request (p. 417).

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <remove_resource>
        <resource_pool xsi:type="ns2:resource_ip_poolType">
          <type>resource_ip</type>
          <ip_range>
            <start_ip>10.17.3.125</start_ip>
            <end_ip>10.17.3.127</end_ip>
          </ip_range>
        </resource_pool>
      </remove_resource>
    </resourcem>
  </data>
</packet>
```

set_pool

Summary:

Deletes all resources from a resource pool and then adds the specified new resources.

Request specification:

Name	Min/Max	Type	Description
`set_pool`	`1..1`		
{			
`resource_pool`	`1..[]`	`resource_poolType` (p. 414)	The new resource pool configuration. Use the appropriate subtype of `resource_poolType` (p. 414) to set the resources for the desired pool.
}			

Returns:

OK/Error

Description:

The call first removes all resources from a given pool, and then adds the new ones. The call does not check if the resources are currently in use, and it does not check if the new values are valid or are, in general, available. If you pass an empty resource structure, the call will remove all resources from a pool without adding the new ones.

Example 1:

The following example removes all current resources from the IP address pool and the adds a new IP address range.

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <set_pool>
        <resource_pool xsi:type="ns2:resource_ip_poolType">
          <type>resource_ip</type>
          <ip_range>
            <start_ip>10.17.3.125</start_ip>
            <end_ip>10.17.3.127</end_ip>
          </ip_range>
        </resource_pool>
      </set_pool>
    </resourcem>
  </data>
</packet>
```

Example 2

The following example removes all resources from the IP address pool without adding any new resources.

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <set_pool>
        <resource_pool xsi:type="ns2:resource_ip_poolType">
          <type>resource_ip</type>
        </resource_pool>
      </set_pool>
    </resourcem>
  </data>
</packet>
```

get_pool

Summary:

Retrieves the list of resources from a resource pool.

Request specification:

Name	Min/Max	Type	Description

get_pool	1..1		
{			
resource_pool	0..[]	resource_poolType (p. 414)	Use this element to specify the type of the resource to get the configuration for. If this element is omitted, returns the configurations of all available pools.
}			

Returns:

Name	Min/Max	Type	Description
resource_pool	0..[]	resource_poolType (p. 414)	A list of resources. The actual data type returned depends on the given resource pool type. See the subtypes of resource_poolType (p. 414) for the available options.

Example:

Retrieving the IP pool configuration.

Input

```
<packet version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <get_pool>
        <resource_pool>
          <type>resource_ip</type>
        </resource_pool>
      </get_pool>
    </resourcem>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="2dc45f980a0t1238r4e4" time="2007-03-12T17:08:02+0000" priority="0"
version="4.0.0">
<ns1:origin>resourcem</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:resourcem>
    <ns2:resource_pool xsi:type="ns2:resource_ip_poolType">
      <ns2:type>resource_ip</ns2:type>
      <ns2:ip_range>
        <ns2:start_ip>10.17.3.125</ns2:start_ip>
        <ns2:end_ip>10.17.3.127</ns2:end_ip>
      </ns2:ip_range>
    </ns2:resource_pool>
  </ns2:resourcem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

allocate

Summary:

Allocates a resource from a resource pool.

Request specification:

Name	Min/Max	Type	Description
allocate	1..1		
{			
resource_pool	0..1	resource_poolType (p. 414)	Use this element to specify the type of the resource pool.

resource	0..1	resourceType	The resource to allocate (e.g. IP address). If this element is omitted, the resource will be allocated automatically from those available in the pool. Use the appropriate subtype of resourceType to match the resource type specified in the resource_pool element.
count	0..1	int	Number of resource items to allocate. If not specified, one item will be allocated.
}			

Returns:

Name	Min/Max	Type	Description
resource	1..1	resourceType (p. 55)	The allocated resource information.

Description:

Use the allocate call to allocate a resource from a resource pool. You can allocate a specific resource (for example, a specific IP address) or you can get the next available resource by omitting the `resource` element. Once the resource is allocated, you can use it in your application. For example, after you allocate an IP address, you can assign it to a server.

Example 1:

The following example shows how to allocate a specific IP address from the IP address pool.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <allocate>
        <resource_pool xsi:type="ns2:resource_ip_poolType">
          <type>resource_ip</type>
        </resource_pool>
        <resource xsi:type="ns2:resource_ipType">
          <ip>10.17.3.127</ip>
        </resource>
      </allocate>
    </resourcem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="3ac45ffded6t6b89r4e4" time="2007-03-15T23:40:55+0000" priority="0"
version="4.0.0">
<ns1:origin>resourcem</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:resourcem>
    <ns2:resource xsi:type="ns2:resource_ipType">
      <ns2:ip>10.17.3.127</ns2:ip>
    </ns2:resource>
  </ns2:resourcem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 2:

Automatically allocating an IP address from the IP address pool.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <allocate>
        <resource_pool xsi:type="ns2:resource_ip_poolType">
          <type>resource_ip</type>
        </resource_pool>
      </allocate>
    </resourcem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="43c45ffe196t22eer4e4" time="2007-03-15T23:48:49+0000" priority="0"
version="4.0.0">
<ns1:origin>resourcem</ns1:origin>
<ns1:target>vzclient8</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:resourcem>
    <ns2:resource xsi:type="ns2:resource_ipType">
      <ns2:ip>10.17.3.126</ns2:ip>
    </ns2:resource>
  </ns2:resourcem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

release

Summary:

Releases an allocated resource.

Request specification:

Name	Min/Max	Type	Description
`release`			
`{`			
`  resource`	`1..[]`	`resourceType (p. 55)`	Resource to release.
`}`			

Returns:

OK/Error

Example:

The following example shows how to release an IP address that was previously allocated from the IP address pool.

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/resourcem"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>resourcem</target>
  <data>
    <resourcem>
      <release>
        <resource xsi:type="ns2:resource_ipType">
          <ip>10.17.3.127</ip>
        </resource>
      </release>
    </resourcem>
  </data>
</packet>
```

scheduler

Purpose:

The `scheduler` interface allows to schedule unattended running of Agent tasks. There's absolutely no restrictions on the type of the task that can be scheduled. Any request that can normally be sent to Agent can also be added to the scheduler to be executed on the specified date at the specified time, or to be executed on a periodic basis.

Types

daily_triggerType

Summary:

This type is used to define a daily schedule policy.

Type specification:

Extends `triggerType (p. 433)`

Adds the following elements:

Name	Min/Max	Type	Description
[			*The following elements constitute a choice group. You can use only one element from the group in a single call, but not two or all three together.*

days_interval	1..1	int	The day interval from the range of 1 to 365. The number here determines the number of days between firings. For example, to execute a task every day, specify 1; to execute a task every other day, specify 2, etc.
workdays	1..1	none	Execute the task every working day of the week (Monday through Friday) at the time specified in the start_time parameter for as many weeks (months, years) as determined by the start_time and end_time parameters.
weekends	1..1	none	Execute the task on Saturdays and Sundays only.
]			

Example:

Execute the task every day at midnight for two months.

```
<start_time>2007-03-01T00:00:00-0500</start_time>
<end_time>2007-05-01T00:00:00-0500</end_time>
<days_interval>1</days_interval>
```

Execute the task every Saturday and Sunday at 16:05 (4:05 pm) for 1 year.

```
<start_time>2007-03-01T16:05:00-0500</start_time>
<end_time>2008-03-01T16:05:00-0500</end_time>
<weekends/>
```

monthly_day_of_week_triggerType

Summary:

This type is used to define a monthly-day-of-week schedule policy.

Type specification:

Extends `triggerType (p. 433)`

Adds the following elements:

Name	Min/Max	Type	Description
`day_of_week`	`1..7`	`none`	The day of week on which to execute the task. The possible values are: `0` -- Sunday `1` -- Monday `2` -- Tuesday `3` -- Wednesday `4` -- Thursday `5` -- Friday `6` -- Saturday You may include as much as 7 occurrence of this element in a single packet. For example, to execute a task every Saturday and Monday, one occurrence should contain 6, and the other should contain 0.

weekday_of_month	1..6	none	The weekday of month on which to execute the task. The possible values are: 1 -- First 2 -- Second 3 -- Third 4 -- Forth 5 -- Fifth 6 -- Last You may include as many as 6 occurrence of this element in a single packet. For example, to execute a task every first and every second day-of-week of the month (for example Friday, the actual day is determined by the day_of_week parameter), one occurrence should contain 1 and the other should contain 2.
month_of_year	1..12	none	The month of the year from the range of 1 to 12 on which to execute the task. You may include as many as 12 occurrences of this element in a single packet. For example, to execute a task every January and December, one occurrence should contain 1 and the other should contain 12.

Examples:

Execute the task at midnight every last Friday of the month, March through May.

```
<start_time>2007-02-01T00:00:00-0500</start_time>
<day_of_week>5</day_of_week>
<weekday_of_month>6</weekday_of_month>
<month_of_year>3</month_of_year>
<month_of_year>4</month_of_year>
<month_of_year>5</month_of_year>
```

monthly_triggerType

Summary:

This type is used to define a monthly schedule policy.

Type specification:

Extends `triggerType (p. 433)`

Adds the following elements:

Name	Min/Max	Type	Description
`day_of_month`	`1..32`	`none`	The day of the month from the range of 1 to 31 (or 32 for the last day of the month) on which to execute the task. You may include as many as 32 occurrences of the element in a single packet, each specifying a day on which to execute a task. For example, to execute a task on the 1st and the last day of the month, one occurrence should contain 1 and the other occurrence should contain 32.
`month_of_year`	`1..12`	`none`	The month on which to execute the task from the range of 1 to 12. You may include as many as 12 occurrences of this element in a single packet, each specifying a month on which the task should be executed. For example, to execute a task every January and December, one occurrence should contain 1 and the other occurrence should contain 12.

Example:

Execute the task on July 4th of every year with no task expiration date.

```
<start_time>2007-03-01T16:05:00-0500</start_time>
<day_of_month>4</day_of_month>
<month_of_year>7</month_of_year>
```

Execute the task on the first and the last day of each month with no task expiration date.

```
<start_time>2007-03-01T16:05:00-0500</start_time>
<day_of_month>1</day_of_month>
<day_of_month>32</day_of_month>
<month_of_year>1</month_of_year>
<month_of_year>2</month_of_year>
<month_of_year>3</month_of_year>
<month_of_year>4</month_of_year>
<month_of_year>5</month_of_year>
<month_of_year>6</month_of_year>
<month_of_year>7</month_of_year>
<month_of_year>8</month_of_year>
<month_of_year>9</month_of_year>
<month_of_year>10</month_of_year>
<month_of_year>11</month_of_year>
<month_of_year>12</month_of_year>
```

once_triggerType

Summary:

This type is used to schedule a task to be executed only once. The date and time at which the task will be run is determined by the `start_time` parameter. The `end_time` parameter is not used here.

Type specification:

Extends `triggerType` (p. 433)

The type has no additional elements.

taskType

Summary:

Defines a scheduler task.

Type specification:

Name	Min/Max	Type	Description
`id`	`0..1`	`guid_type` (p. 26)	Task ID. The value is initially generated by Agent when the task is added to the scheduler. You will need this ID to perform other task operations, such as removing, listing, or updating the task.
`title`	`1..1`	`base64Binary`	Task title.

description	0..1	base64Binary	Task description.
category	1..1	string	Task category. This parameter is used by Virtuozzo Tools to identify the task category. You may use any value that you like here.
triggers	1..1		Schedule policy.
{			
trigger	1..1	triggerType (p. 433)	This parameter determines how and when the task will be executed. Use the appropriate subtype of the triggerType depending on the schedule policy type (i,e, daily, weekly, etc.)
}			
next_start	0..1	datetime_type (p. 25)	This is a read-only parameter. Do not use it in the add (p. 435) or update (p. 439) calls. Indicates the date and time of the next firing according to the current schedule.
packet	0..1	base64Binary	This is the actual task, a base-64-encoded string containing a valid Agent XML message that will be executed according to the schedule specified in the triggers parameter (above).
disabled	0..1	boolean	Indicates whether the task is disabled or enabled. true -- the task is disabled. false or absent -- the task is enabled. You may use this parameter in the update (p. 439) call to enable or disable an existing task.

triggerType

Summary:

This is the base type defining the schedule policy. When adding a task to the scheduler, choose from one of the subtypes of this type depending on how and when you would like the task to be executed. See the sections that describe the subtypes of this type for the detailed descriptions and code example on how to compose a schedule.

Type specification:

Name	Min/Max	Type	Description
`start_time`	`1..1`	`datetime_type` (p. 25)	The date and time of a scheduler task activation. The date portion is not necessarily when the trigger will fire. The actual firing date is determined by other schedule policy parameters. The time portion determines the exact time when the task will be executed on the scheduled day(s).
`end_time`	`0..1`	`datetime_type` (p. 25)	The date and time of a scheduler task deactivation. If absent, the trigger will stay active indefinitely until it is disabled using the `update` call (p. 439) or removed from the scheduler using the `remove` call (p. 437).

Subtypes:

`once_triggerType` (p. 431)

`daily_triggerType` (p. 426)

`weekly_triggerType` (p. 434)

`monthly_triggerType` (p. 430)

`monthly_day_of_week_triggerType` (p. 428)

weekly_triggerType

Summary:

This type is used to define a weekly schedule policy.

Type specification:

Extends `triggerType (p. 433)`

Adds the following elements:

Name	Min/Max	Type	Description
`weeks_interval`	`1..1`	`int`	The week interval from the range of 1 to 52. The number here determines the number of weeks between firings. For example, to execute a task every week, specify 1; to execute a task every other week, specify 2, etc.
`day_of_week`	`1..7`	`none`	The days of the week to execute the task on. The possible values are as follows: `0` -- Sunday `1` -- Monday `2` -- Tuesday `3` -- Wednesday `4` -- Thursday `5` -- Friday `6` -- Saturday You may include as many as 7 occurrences of this element in a single packet, each specifying a day of the week on which the task should be execute. For example, to execute the task on every Monday and Wednesday, one occurrence should contain 1 and the other occurrence should contain 3.

Example:

Execute a task at midnight on every Monday, Wednesday, and Friday, every other week for 1 year.

```
<start_time>2007-03-01T00:00:00-0500</start_time>
<end_time>2008-03-01T00:00:00-0500</end_time>
<weeks_interval>2</weeks_interval>
<day_of_week>1</day_of_week>
<day_of_week>3</day_of_week>
<day_of_week>5</day_of_week>
```

Calls

Call	Description
add (p. 435)	Adds a task to the scheduler.
remove (p. 437)	Removes a task from the scheduler.
list (p. 437)	Lists existing tasks.
update (p. 439)	Updates an existing task.

add

Summary:

Adds a task to the scheduler.

Request specification:

Name	Min/Max	Type	Description
`add`	`1..1`		
`{`			
`  task`	`1..1`	`taskType` (p. 431)	Task definition.
`}`			

Returns:

Name	Min/Max	Type	Description
`task`	`1..1`	`taskType` (p. 431)	The ID that was assigned to the task.

Description:

See `taskType` (p. 431) for the descriptions and examples of how to schedule a task.

It is possible to set the maximum allowable number of simultaneously scheduled tasks, which is usually done in order to avoid scheduler overflow. The number is defined in the Agent configuration. The parameter name is `max_tasks_count`, which is located in the `scheduler/configuration` section.

Example:

Scheduling to perform a server backup on a daily basis. The backups will be performed at midnight on every day of the week, including weekdays and weekends. The backups will begin on March 03, 2007 and will end on May 05, 2007.

Input

```
<packet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>scheduler</target>
  <data>
    <scheduler>
      <add>
        <task>
          <title>Test-Backup</title>
          <triggers>
            <trigger xsi:type="daily_triggerType">
              <start_time>2007-03-01T00:00:00-0500</start_time>
              <end_time>2007-05-01T00:00:00-0500</end_time>
              <days_interval>1</days_interval>
            </trigger>
          </triggers>

<packet>PHBhY2tldCB2ZXJzaW9uPSI0LjAuMCI+PHRhcmdldD5iYWNrdXBtPC90YXJnZXQ+PGRhdG
E+PGJhY2t1cG0+PGJhY2t1cF9lbnY+PGVudl9saXN0PjxlaWQ+NTdjMmNkNmMtYzAyYi00NjQ1LWJk
YjUtZTg4M2VhMDA1ODk2PC9laWQ+PC9lbnZfbGlzdD48YmFja3VwX29wdGlvbnM+PHR5cGU+MDwvdH
lwZT48Y29tcHJlc3Npb24+MjwvY29tcHJlc3Npb24+PGRlc2NyaXB0aW9uPlJuVnNiQ0JpWVdOcmRY
QWdNakF3Tnkwd01TMHhNZz09PC9kZXNjcmlwdGlvbj48L2JhY2t1cF9vcHRpb25zPjxiYWwvPj
wvYmFja3VwX2Vudj48L2JhY2t1cG0+PC9kYXRhPjwvcGFja2V0Pg==</packet>
        </task>
      </add>
    </scheduler>
  </data>
</packet>
```

Output

The output contains the task ID.

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/scheduler"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="26c45ed906at428br274" time="2007-03-04T14:55:22+0000" priority="0"
version="4.0.0">
<ns1:origin>scheduler</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:scheduler>
    <ns2:id>f40d0ba4-e561-4d4a-a1f4-01e0156a91f5</ns2:id>
  </ns2:scheduler>
</ns1:data>
```

```
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

remove

Summary:

Removes an existing task from scheduler.

Request specification:

Name	Min/Max	Type	Description
remove	1..1		
{			
id	1..1	guid_type (p. 26)	Task ID. Use the list call (p. 437) to get a list of the existing tasks.
}			

Returns:

OK/Error

Description:

Removes a task from scheduler that was previously added with the add request (p. 435). This operation is irreversible. Once the task is removed, it cannot be recovered. To temporarily disable the task, use the update call (p. 439) specifying the disabled parameter.

Example:

```
<packet>
  <target>scheduler</target>
  <data>
    <scheduler>
      <remove>
        <task>
          <id>da696f91-887e-460e-bd42-93f161f6afc6</id>
        </task>
      </remove>
    </scheduler>
  </data>
</packet>
```

list

Summary:

Lists existing tasks.

Request specification:

Name	Min/Max	Type	Description
list	1..1		

{			
id	0..1	guid_type (p. 26)	Task ID. If this element is omitted, the call returns all existing tasks.
}			

Returns:

Name	Min/Max	Type	Description
task	1..[]	taskType (p. 431)	Task list.

Example:

Input

```
<packet version="4.0.0">
  <target>scheduler</target>
  <data>
    <scheduler>
      <list/>
    </scheduler>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/scheduler"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="6c45e6d45ft2cd6rf2c"
time="2007-02-28T12:36:31+0000" priority="0" version="4.0.0">
<ns1:origin>scheduler</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:scheduler>
    <ns2:task>
      <ns2:id>f40d0ba4-e561-4d4a-a1f4-01e0156a91f5</ns2:id>
      <ns2:title>Test-Backup</title>
        <ns2:triggers>
          <ns2:trigger xsi:type="daily_triggerType">
            <ns2:start_time>2007-03-01T00:00:00-0500</ns2:start_time>
            <ns2:end_time>2007-05-01T00:00:00-0500</ns2:end_time>
            <ns2:days_interval>1</ns2:days_interval>
          </ns2:trigger>
        </ns2:triggers>

<ns2:packet>PHBhY2tldCB2ZXJzaW9uPSI0LjAuMCI+PHRhcmdldD5iYWNrdXBtPC90YXJnZXQ+PG
RhdGE+PGJhY2t1cG0+PGJhY2t1cF9lbnY+PGVudl9saXN0PjxlaWQ+NTdjMmNkNmMtYzAyYi00NjQ1
LWJkYjUtZTg4M2VhMDA1ODk2PC9laWQ+PC9lbnZfbGlzdD48YmFja3VwX29wdGlvbnM+PHR5cGU+MD
wvdHlwZT48Y29tcHJlc3Npb24+MjwvY29tcHJlc3Npb24+PGRlc2NyaXB0aW9uPlJuVnNiQ0JpWVdO
cmRYQWdNakF3Tnkwd01TMHhNZz09PC9kZXNjcmlwdGlvbj48L2JhY2t1cF9vcHRpb25zPjwvZW5iYW
wvPjwvYmFja3VwX2Vudj48L2JhY2t1cG0+PC9kYXRhPjwvZGFja2V0Pg==</ns2:packet>
    </ns2:task>
  </ns2:scheduler>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

update

Summary:

Updates an existing task.

Request specification:

Name	Min/Max	Type	Description
`update`	`1..1`		
`{`			
`  task`	`1..1`	`taskType` (p. 431)	Task definition.
`}`			

Returns:

OK/Errors.

Description:

The `update` call completely removes the specified task first and then inserts the new values, thus completely replacing the existing task (the task ID is preserved, however). If you would like to update a specific parameter (for example, to disable a task or to change its schedule policy), you will first have to retrieve the entire task definition using the `list` call (p. 437). Once you have the original task data, update the parameters in this structure as needed, and then pass the updated structure to the `update` call.

Example:

```
<packet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>scheduler</target>
  <data>
    <scheduler>
      <update>
        <task>
          <id>f40d0ba4-e561-4d4a-a1f4-01e0156a91f5</id>
          <title>Test-Backup</title>
            <triggers>
              <trigger xsi:type="daily_triggerType">
                <start_time>2007-03-01T00:00:00-0500</start_time>
                <end_time>2007-05-01T00:00:00-0500</end_time>
                <days_interval>1</days_interval>
              </trigger>
            </triggers>
           <packet>The actual task (XML) goes here</packet>
           <disabled>true</disabled>
        </task>
      </update>
    </scheduler>
  </data>
</packet>
```

server_group

Purpose:

The Virtuozzo group management interface.

Virtuozzo group is a collection of servers running Agent software, interconnected using internal Agent mechanisms for the purpose of providing integrated access to their resources. A Virtuozzo group consists of a Master Node and one or more Slave Nodes. The Master Node is a computer that the users interact with. It provides access to other computers in the group transparently to the user. The Master Node manages the entire group by allocating and controlling the available resources. The `server_group` interface allows to set up and administer Virtuozzo groups.

Once a Virtuozzo group is set up, all of the servers in the group become one logical computing unit with a shared resource pool and a common management interface. The following is a list of typical management tasks performed in a Virtuozzo group:

- Retrieve a list of all Virtuozzo Containers from every server in a single call. Each Container has a globally unique ID assigned to it by Agent, so the list will always contain unique entries.
- Perform any of the usual management tasks on any of the Containers in the entire group. A client program connected to the Master Node can manage Virtuozzo Containers without even knowing their actual hosts. However, the Server ID of any host (Slave Node) can be easily obtained at any time if needed.
- Get a combined list of all of the OS and application templates available in the group. The templates can be copied from one node to another and deployed to multiple nodes.
- Get a combined list of all the sample configurations available in the group. The configurations can then be used to create Virtuozzo Containers on any node in the group.
- Get a combined list of the Virtuozzo Container backups available in the group. A backup can be restored to any node in the group.

Types

configType

Summary:

Virtuozzo group network configuration information.

Type specification:

Name	Min/Max	Type	Description
nameservers	0..1		Nameserver information.
{			
nameserver	0.[]	ip_type	Nameserver IP addresses.
}			
search_domains	0..1		Search domain information.
{			
search_domain	0..[]	string	Search domain names.
}			
networks			Virtuozzo virtual network information.
{			

network	0..[]	networkType (p. 442)	A list of the available Virtuozzo virtual networks. This field is used by Virtuozzo Tools as an easy way of listing the virtual networks available in a given Virtuozzo group. Setting the network ID through this field does not actually create a network. The network must be created through the vzanetworkm interface (p. 606).
}			

networkType

Summary:

Contains Virtuozzo virtual network ID and description.

Type specification:

Name	Min/Max	Type	Description
id	1..1	base64Binary	Virtuozzo virtual network ID.
description	0..1	base64Binary	Virtual network description.

voc_idType

Summary:

Contains the Agent vocabulary name and version.

Type specification:

Name	Min/Max	Type	Description
name	1..1	string	Vocabulary name.
version	1..1	string	Vocabulary version.

Calls

Call	Description
add (p. 443)	Adds a slave node to a Virtuozzo group.
del (p. 445)	Deletes a slave node from a Virtuozzo group.

get_list (p. 445)	Retrieves a list of servers from a Virtuozzo group.
get_info (p. 448)	Returns information about the specified Container.
destroy (p. 453)	Dismantles a Virtuozzo group.
set_config (p. 454)	Modifies the Virtuozzo group configuration.
get_config (p. 454)	Retrieves the VIrtuozzo group configuration information.
get_env_voc (p. 456)	Retrieves Agent vocabularies.
get_vocabulary (p. 456)	Retrieves the specified vocabulary data.

add

Summary:

Adds a Hardware Node to a Virtuozzo group.

Request specification:

Name	Min/Max	Type	Description
add			
{			
conn_info	1..1	connection_infoType (p. 30)	The Slave Node connection information.
master	0..1	connectivity_infoType **(p. 30)**	The Master Node connection information. This parameter should be used when master node has more than one IP address. For example, this parameter should be set when the master and the Slave Nodes are located in the same private network and should communicate using IP addresses other than those exposed to the external network. If Master Node has more than one IP address but this parameter is not set, the Master will select the address randomly.
force	0..1	none	Forcibly add a Node to the group even if it's already registered with another group.
}			

Returns:

Name	Min/Max	Type	Description
eid	1..1	eid_type (p. 25)	The Server ID of the Node that you just added to the Virtuozzo group.

Description:

The call adds a Slave Node to the Virtuozzo group by registering it with the Master Node. The call must be executed on the Master Node.

The server_group interface does not have a separate call that creates a Virtuozzo group. A group is created and configured automatically as soon as you execute at least one successful add request. The Node on which you execute the call becomes the Master Node. To add more Nodes to the Group, execute the add request for each one of them. There can be only one Master Node in a Virtuozzo group.

Example:

Input

```
<packet version="4.0.0">
  <target>server_group</target>
  <data>
    <server_group>
      <add>
        <conn_info>
          <protocol>SSL</protocol>
          <address>192.168.0.250</address>
          <login>
            <name>dnphZG1pbg==</name>
          </login>
          <password>MXEzNl</password>
          <port>4434</port>
        </conn_info>
      </add>
    </server_group>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="ec45d1b013t26e9rd54"
time="2007-02-12T06:26:17+0000" priority="0" version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns1:ok/>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

del

Summary:

Removes a Slave Node from a Virtuozzo group.

Request specification:

Name	Min/Max	Type	Description
del			
{			
eid	1..1	eid_type (p. 25)	Server ID of the slave Node being removed from the group.
force	0..1	none	Forcibly remove the Node.
}			

Returns:

OK/Error

Description:

The call removes a slave Node from a Virtuozzo group. The request must be executed on the Master Node of the group. As a result, the specified slave Node will no longer be registered with this group and can be added to a different group or become a Master Node of a new group.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>server_group</target>
  <data>
    <server_group>
      <del>
        <eid>15b0b336-1a53-4f5e-8e15-19ba4a2dbd9d</eid>
      </del>
    </server_group>
  </data>
</packet>
```

get_list

Summary:

Retrieves a list of servers from a Virtuozzo group.

Request specification:

Name	Min/Max	Type	Description
get_list			
{			

count	0..1	int	Maximum number of servers to include in the list. Omit the element to retrieve all available servers.
type	0..1	string	Types of servers to retrieve: generic -- physical servers. virtuozzo -- Virtuozzo Containers. Omit the element to retrieve the servers of all known types.
status	0..[]	env_statusType (p. 34)	Retrieve only the servers with the specified status.
options	0..1		Additional search options.
{			
hostname	0..1	string	Hostname.
ip	0..1	ip_type (p. 26)	IP address.
}			
}			

Returns:

Name	Min/Max	Type	Description
eid	0..[]	eid_type (p. 25)	The list of Server IDs.

Description:

The call retrieves a list of IDs of the servers from a Virtuozzo group. Use the provided options to specify a search criteria to retrieve only the servers that you need. The call must be executed on the Master Node.

Example 1:

The following example retrieves a list of the physical servers, including Master Node and all Slave Nodes.

Input

```
<packet version="4.0.0" id="2">
  <target>server_group</target>
  <data>
    <server_group>
      <get_list>
        <type>generic</type>
      </get_list>
    </server_group>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="17c45d1c4f4t99rd54"
time="2007-02-12T07:35:39+0000" priority="0" version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns2:eid>89e27960-97b8-461f-902f-557b4b16784b</ns2:eid>
    <ns2:eid>72145bf0-7562-43d4-b707-cc33d37e3f10</ns2:eid>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Example 2:

The following example retrieves a list of all servers, including physical servers and Virtuozzo Containers.

Input

```
<packet version="4.0.0" id="2">
  <target>server_group</target>
  <data>
    <server_group>
      <get_list/>
    </server_group>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="14c45d1c7cbt7e87r538" time="2007-02-12T05:13:48+0000" priority="0"
version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns2:eid>89e27960-97b8-461f-902f-557b4b16784b</ns2:eid>
    <ns2:eid>3288bb6b-8a49-4230-b565-6ad5521182aa</ns2:eid>
    <ns2:eid>57c2cd6c-c02b-4645-bdb5-e883ea005896</ns2:eid>
    <ns2:eid>72145bf0-7562-43d4-b707-cc33d37e3f10</ns2:eid>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_info

Summary:

Retrieves information about the specified server in a Virtuozzo group.

Request specification:

Name	Min/Max	Type	Description
get_info			
{			
eid	0..[]	eid_type (p. 25)	Server ID. If omitted, then retrieves information for all servers in a group.
status	0..1	none	If present, a server status information will be included in the result.
config	0..1	none	If present, a server configuration information will be included in the result.
filter_config	0..1		Filter. Allows to perform selective retrieval.
{			
<xs:any/>	1..1		Specify server configuration parameters that you would like to be retrieved.
}			
}			

Returns:

Name	Min/Max	Type	Description
env	0..[]	envType	The requested server information.

Example:

Input

```
<packet version="4.0.0">
  <target>server_group</target>
  <data>
    <server_group>
      <get_info>
        <eid>3288bb6b-8a49-4230-b565-6ad5521182aa</eid>
        <status/>
        <config/>
      </get_info>
    </server_group>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:ns4="http://www.swsoft.com/webservices/vza/4.0.0/vzatypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="25c45d1d304t66bbr538" time="2007-02-12T05:51:03+0000" priority="0"
version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns2:env xsi:type="ns3:envType">
      <ns3:parent_eid>89e27960-97b8-461f-902f-557b4b16784b</ns3:parent_eid>
      <ns3:eid>3288bb6b-8a49-4230-b565-6ad5521182aa</ns3:eid>
      <ns3:status xsi:type="ns3:env_statusType">
        <ns3:state>6</ns3:state>
      </ns3:status>
      <ns3:alert>0</ns3:alert>
      <ns3:config xsi:type="ns3:env_configType"/>
      <ns3:virtual_config xsi:type="ns3:venv_configType">
        <ns3:hostname>Host-105</ns3:hostname>
        <ns3:name>Test-VE5</ns3:name>
        <ns3:offline_management>1</ns3:offline_management>
        <ns3:on_boot>1</ns3:on_boot>
        <ns3:os_template>
          <ns3:version>20061020</ns3:version>
          <ns3:name>redhat-as3-minimal</ns3:name>
        </ns3:os_template>
        <ns3:ve_root>/vz/root/$VEID</ns3:ve_root>
        <ns3:ve_private>/vz/private/$VEID</ns3:ve_private>
        <ns3:qos>
          <ns3:id>avnumproc</ns3:id>
          <ns3:hard>40</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>cpuunits</ns3:id>
          <ns3:hard>1000</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>dcachesize</ns3:id>
          <ns3:hard>1097728</ns3:hard>
          <ns3:soft>1048576</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>dgramrcvbuf</ns3:id>
```

```
          <ns3:hard>132096</ns3:hard>
          <ns3:soft>132096</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>diskinodes</ns3:id>
          <ns3:hard>220000</ns3:hard>
          <ns3:soft>200000</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>diskspace</ns3:id>
          <ns3:hard>1153434</ns3:hard>
          <ns3:soft>1048576</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>kmemsize</ns3:id>
          <ns3:hard>2936012</ns3:hard>
          <ns3:soft>2752512</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>lockedpages</ns3:id>
          <ns3:hard>32</ns3:hard>
          <ns3:soft>32</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numfile</ns3:id>
          <ns3:hard>2048</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numflock</ns3:id>
          <ns3:hard>110</ns3:hard>
          <ns3:soft>100</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numiptent</ns3:id>
          <ns3:hard>128</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numothersock</ns3:id>
          <ns3:hard>80</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numproc</ns3:id>
          <ns3:hard>65</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numpty</ns3:id>
          <ns3:hard>16</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numsiginfo</ns3:id>
          <ns3:hard>256</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>numtcpsock</ns3:id>
          <ns3:hard>80</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>oomguarpages</ns3:id>
          <ns3:hard>2147483647</ns3:hard>
          <ns3:soft>6144</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>othersockbuf</ns3:id>
          <ns3:hard>336896</ns3:hard>
          <ns3:soft>132096</ns3:soft>
        </ns3:qos>
        <ns3:qos>
```

```
          <ns3:id>physpages</ns3:id>
          <ns3:hard>2147483647</ns3:hard>
          <ns3:soft>0</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>privvmpages</ns3:id>
          <ns3:hard>24576</ns3:hard>
          <ns3:soft>22528</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>quotatime</ns3:id>
          <ns3:hard>0</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>shmpages</ns3:id>
          <ns3:hard>8192</ns3:hard>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>tcprcvbuf</ns3:id>
          <ns3:hard>524288</ns3:hard>
          <ns3:soft>319488</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>tcpsndbuf</ns3:id>
          <ns3:hard>524288</ns3:hard>
          <ns3:soft>319488</ns3:soft>
        </ns3:qos>
        <ns3:qos>
          <ns3:id>vmguarpages</ns3:id>
          <ns3:hard>2147483647</ns3:hard>
          <ns3:soft>6144</ns3:soft>
        </ns3:qos>
        <ns3:veid>105</ns3:veid>
        <ns3:type>virtuozzo</ns3:type>
        <ns3:offline_service>vzpp</ns3:offline_service>
        <ns3:offline_service>vzpp-plesk</ns3:offline_service>
        <ns3:os xsi:type="ns3:osType">
          <ns3:platform>Linux</ns3:platform>
          <ns3:kernel>2.6.9-023stab033.6</ns3:kernel>
          <ns3:version>20061020</ns3:version>
          <ns3:name>redhat-as3-minimal</ns3:name>
        </ns3:os>
        <ns3:net_device xsi:type="ns4:net_vethType">
          <ns3:id>venet0</ns3:id>
          <ns3:ip_address>
            <ns3:ip>10.130.1.5</ns3:ip>
          </ns3:ip_address>
          <ns4:host_routed/>
        </ns3:net_device>
        <ns3:address>
          <ns3:ip>10.130.1.5</ns3:ip>
        </ns3:address>
      </ns3:virtual_config>
    </ns2:env>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

destroy

Summary:

Dismantles an existing VIrtuozzo group.

Request specification:

Name	Min/Max	Type	Description
destroy	1..1	none	

Returns:

OK/Error

Description:

The call dismantles a Virtuozzo group by removing all relevant references from the Master and Slave Nodes. Once completed, the Nodes may be added to other Virtuozzo groups. Any former participant of a group may also become a Master Node forming its own group. The call must be executed on the Master Node.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>server_group</target>
  <data>
    <server_group>
      <destroy/>
    </server_group>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="26c45d1d434t428br538" time="2007-02-12T05:54:53+0000" priority="0"
version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns1:ok/>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

set_config

Summary:

Modifies Virtuozzo Group configuration.

Request specification:

Name	Min/Max	Type	Description
set_config	1..1	none	
{			
config	1..1	configType (p. 441)	Virtuozzo Group configuration.
}			

Returns:

OK/Error

Example:

Adding two name servers and one search domain to a Virtuozzo Group. The call must be executed on the Master Node.

```
<packet version="4.0.0">
  <target>server_group</target>
  <data>
    <server_group>
      <set_config>
        <config>
          <nameservers>
            <nameserver>192.168.1.51</nameserver>
            <nameserver>192.168.1.52</nameserver>
          </nameservers>
          <search_domains>
            <search_domain>ts6.com</search_domain>
          </search_domains>
        </config>
      </set_config>
    </server_group>
  </data>
</packet>
```

get_config

Summary:

Retrieves configuration information for a given Virtuozzo Group.

Request specification:

Name	Min/Max	Type	Description
get_config			

Returns:

Name	Min/Max	Type	Description
config	0..1	configType (p. 441)	The cluster network configuration information.

Example:

Retrieving Virtuozzo Group configuration information. The call must be executed on the Master Node.

Input

```
<packet version="4.0.0">
  <target>server_group</target>
  <data>
    <server_group>
      <get_config/>
    </server_group>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/server_group"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="29c45d1e0ebt5d03r538" time="2007-02-12T06:35:58+0000" priority="0"
version="4.0.0">
<ns1:origin>server_group</ns1:origin>
<ns1:target>vzclient4</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:server_group>
    <ns2:config>
      <ns2:nameservers>
        <ns2:nameserver>192.168.1.51</ns2:nameserver>
        <ns2:nameserver>192.168.1.52</ns2:nameserver>
      </ns2:nameservers>
      <ns2:search_domains>
        <ns2:search_domain>ts6.com</ns2:search_domain>
      </ns2:search_domains>
      <ns2:networks/>
    </ns2:config>
  </ns2:server_group>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

get_env_voc

Summary:

Retrieves a list of vocabularies from a given Virtuozzo Group.

Request specification:

Name	Min/Max	Type	Description
eid	0..[]	eid_type (p. 25)	The IDs of the servers to retrieve the vocabularies from. If omitted, a combined list of vocabularies from all servers in the group will be retrieved.

Returns:

Name	Min/Max	Type	Description
env_voc			Vocabulary information.
{			
eid	1..1	eid_type (p. 25)	The ID of the servers from which the vocabularies were retrieved.
id	0..[]	voc_idType (p. 442)	A list of vocabularies. Contains vocabulary name and version information.
}			

Example:

Input

```
<packet version="4.0.0">
  <target>server_group</target>
  <data>
    <server_group>
      <get_env_voc/>
    </server_group>
  </data>
</packet>
```

get_vocabulary

Summary:

Retrieves the specified Agent vocabulary.

Request specification:

Name	Min/Max	Type	Description
get_vocabulary			
{			

id	1..[]	voc_idType (p. 442)	Vocabulary ID and version information. To get the list of vocabulary IDs, use the get_env_voc call (p. 456).
}			

Returns:

Name	Min/Max	Type	Description
vocabulary	1..[]	vocabularyType (p. 62)	Vocabulary data.

servicem

Purpose:

The servicem interface allows to manage operating system services.

Types

service_actionType

Summary:

The service_actionType structure contains the basic service properties.

Type specification:

Name	Min/Max	Type	Description
service	1..1		
{			
name	1..1	string	Service name.
level	0..[]	byte	Service run levels.
xinetd	0..1	none	Indicates that this is a xinetd service.
}			

serviceType

Summary:

The `serviceType` structure contains the extended service information.

Type specification:

Name	Min/Max	Type	Description
`service`			
`{`			
`name`	`1..1`	`string`	Service name.
`display_name`	`0..1`	`base64Binary`	Display name.
`level`	`0..[]`	`byte`	Run levels.
`state`	`0..1`	`boolean`	Service state: `true` -- started `false` -- stopped
`readonly`	`0..1`	`none`	Indicates that it is not recommended to change any settings for this service.
`xinetd`	`0..1`	`none`	Indicates that the service is managed by xinetd service.
`description`	`0..1`	`base64Binary`	Service description.
`status`	`0..1`	`string`	Service status (started, stopped, paused).
`startup_type`	`0..1`	`string`	Startup type (manual, automatic, disabled).
`logon_as`	`0..1`	`string`	The logon account assigned to the service.
`dependent`	`0..[]`	`string`	A list of services that depend on this service.
`depended_on`	`0..[]`	`string`	A list of services that this service depends on.
`}`			

Calls

Call	Description
get (p. 459)	Retrieves service information.
set (p. 462)	Sets service levels.
start (p. 463)	Starts a service.

stop (p. 464)	Stops a service.
restart (p. 465)	Restarts a service.
set_startup_type (p. 466)	Sets the startup type for a service.

get

Summary:

Retrieves information about system services.

Request specification:

Name	Min/Max	Type	Description
`get`			
`{`			
`name`	`0..1`	`string`	The name of the service to retrieve the information for. If absent, the information for all services will be retrieved.
`state`	`0..1`	`none`	If present, current service state will be retrieved.
`level`	`0..1`	`none`	If present, the service levels will be retrieved.
`dependencies`	`0..1`	`none`	If present, the service dependencies information will be retrieved.
`}`			

Returns:

Name	Min/Max	Type	Description
`service`	`1..[]`	`serviceType (p. 458)`	Service information.

Description:

The call retrieves a list of services from the current server. You can retrieve the information for a particular service or for all available services. You can also control which service properties will be retrieved by supplying the appropriate parameters. Retrieving such properties as state, level, and dependencies can be a time consuming operation. Please keep that in mind when retrieving a complete list of services. The best practice would be to retrieve just the base service information (all optional parameters are omitted) and then to retrieve the details for each service individually when needed.

The `readonly` element, if present, indicates that the service has the highest possible severity level. It is not recommended to change the settings for such a service. The element will also be present if there's no entry in the Agent vocabulary for this service (i.e. Agent is not aware of it), in which case it is up to you to decide how to handle it.

Example 1:

Retrieving the `TlntSvr` (Telnet) service details, including state, and dependencies from the specified server.

Input

```
<packet version="4.0.0">
  <dst>
    Host74ee4ead-577a-438d-a22b-978922ecdac0</host>
  </dst>
  <target>servicem</target>
  <data>
    <servicem>
      <get>
        <name>TlntSvr</name>
        <state/>
        <dependencies/>
      </get>
    </servicem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/servicem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="6c460004b2t2cd6rd7c" time="2007-03-20T15:59:13+0000" priority="0"
version="4.0.0">
<ns1:origin>74ee4ead-577a-438d-a22b-978922ecdac0</ns1:origin>
<ns1:dst>
  <director>gend</director>
  <target>vzclient2</target>
</ns1:dst>
<ns1:data>
  <ns2:servicem>
    <ns2:service>
      <ns2:name>TlntSvr</ns2:name>
```

```
      <ns2:readonly/>
<ns2:description>RW5hYmxlcyBhIHJlbW90ZSB1c2VyIHRvIGxvZyBvbiB0byB0aGlzIGNvbXB1d
GVyIGFuZCBydW4gcHJvZ3JhbXMsIGFuZCBzdXBwb3J0cyB2YXJpb3VzIFRDUC9JUCBUZWxuZXQgY2x
pZW50cywgaW5jbHVkaW5nIFVOSVgtYmFzZWQgYW5kIFdpbmRvd3MtYmFzZWQgY29tcHV0ZXJzLiBJZ
iB0aGlzIHNlcnZpY2UgaXMgc3RvcHBlZCwgcmVtb3RlIHVzZXIgYWNjZXNzIHRvIHByb2dyYW1zIG1
pZ2h0IGJlIHVuYXZhaWxhYmxlLiBJZiB0aGlzIHNlcnZpY2UgaXMgZGlzYWJsZWQsIGFueSBzZXJ2a
WNlcyB0aGF0IGV4cGxpY2l0bHkgZGVwZW5kIG9uIGl0IHdpbGwgZmFpbCB0byBzdGFydC4=</ns2:d
escription>
      <ns2:state>0</ns2:state>
      <ns2:display_name>VGVsbmV0</ns2:display_name>
      <ns2:status>Stopped</ns2:status>
      <ns2:startup_type>Disabled</ns2:startup_type>
      <ns2:logon_as>NT AUTHORITY\LocalService</ns2:logon_as>
      <ns2:depended_on>RPCSS</ns2:depended_on>
      <ns2:depended_on>TCPIP</ns2:depended_on>
      <ns2:depended_on>NTLMSSP</ns2:depended_on>
    </ns2:service>
  </ns2:servicem>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>servicem</ns1:target>
</ns1:src>
<ns1:target>vzclient2</ns1:target>
</ns1:packet>
```

Example 2:

Retrieving the `crond` service details, including states and levels from a server.

Input

```
<packet version="4.0.0">
  <target>servicem</target>
  <data>
    <servicem>
      <get>
        <name>crond</name>
        <state/>
        <level/>
      </get>
    </servicem>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/servicem"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="2c45ffff73t18ber98c"
time="2007-03-16T01:19:18+0000" priority="0" version="4.0.0">
<ns1:origin>servicem</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns2:servicem>
    <ns2:service>
      <ns2:name>crond</ns2:name>
      <ns2:readonly/>
<ns2:description>Y3JvbiBpcyBhIHN0YW5kYXJkIFVOSVggcHJvZ3JhbSB0aGF0IHJ1bnMgdXNlc
i1zcGVjaWZpZWQgIHByb2dyYW1zIGF0IHBlcmlvZGljIHNjaGVkdWxlZCB0aW1lcy4gdml4aWUgY3J
vbiBhZGRzIGEgIG51bWJlciBvZiBmZWF0dXJlcyB0byB0aGUgYmFzaWMgVU5JWCBjcm9uLCBpbmNsd
WRpbmcgYmV0dGVyICBzZWN1cml0eSBhbmQgbW9yZSBwb3dlcmZ1bCBjb25maWd1cmF0aW9uIG9wdGl
vbnMu</ns2:description>
```

```
      <ns2:state>1</ns2:state>
      <ns2:level>2</ns2:level>
      <ns2:level>3</ns2:level>
      <ns2:level>4</ns2:level>
      <ns2:level>5</ns2:level>
    </ns2:service>
  </ns2:servicem>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

set

Summary:

Sets run levels for the specified services.

Request specification:

Name	Min/Max	Type	Description
set	1..[]	service_actionType (p. 457)	

Returns:

OK/Error.

Description:

This call sets run levels for the specified services. The call is executed internally in two stages:

1 The current run levels are erased.

2 The new run levels are set.

If a service doesn't exist, an error will be returned.

Example:

Setting run level 2 for `httpd` service in the specified server.

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>servicem</target>
  <data>
    <servicem>
      <set>
        <service>
          <name>httpd</name>
          <level>2</level>
        </service>
      </set>
    </servicem>
  </data>
</packet>
```

start

Summary:

Starts the specified services.

This is a logged operation.

Request specification:

Name	Min/Max	Type	Description
`start`	`1..[]`	`service_actionType (p. 457)`	

Returns:

OK/Error.

Description:

The call starts the specified service or services. If a service doesn't exist or cannot be started, an error will be returned.

Example:

Starting the `Telnet` service in the specified server.

```
<packet version="4.0.0">
  <dst>
    Host74ee4ead-577a-438d-a22b-978922ecdac0</host>
  </dst>
  <target>servicem</target>
  <data>
    <servicem>
      <start>
        <service>
          <name>TlntSvr</name>
        </service>
      </start>
    </servicem>
  </data>
</packet>
```

stop

Summary:

Stops the specified services.

A logged operation.

Request specification:

Name	Min/Max	Type	Description
`stop`	`1..[]`	`service_actionType (p. 457)`	
`force`	`0..1`	`none`	If present, the services that depend on this service will be stopped.

Returns:

OK/Error

Description:

The call stops the specified service or services. If a service doesn't exist or cannot be stopped, an error will be returned.

Example:

Stopping the `Telnet` service in the specified server.

```
<packet version="4.0.0">
  <dst>
    Host74ee4ead-577a-438d-a22b-978922ecdac0</host>
  </dst>
  <target>servicem</target>
  <data>
    <servicem>
      <stop>
        <service>
          <name>TlntSvr</name>
        </service>
      </stop>
    </servicem>
  </data>
</packet>
```

restart

Summary:

Restarts the specified services.

A logged operation.

Request specification:

Name	Min/Max	Type	Description
`restart`	`1..[]`	`service_actionType (p. 457)`	

Returns:

OK/Error.

Description:

The call restarts the specified services. If a service doesn't exist or cannot be stopped or started, an error will be returned.

Example:

Restarting the `Telnet` service in the specified server.

```
<packet version="4.0.0">
  <dst>
    Host74ee4ead-577a-438d-a22b-978922ecdac0</host>
  </dst>
  <target>servicem</target>
  <data>
    <servicem>
      <restart>
        <service>
          <name>TlntSvr</name>
        </service>
      </restart>
    </servicem>
  </data>
</packet>
```

set_startup_type

Summary:

Sets startup type for the specified service.

Request specification:

Name	Min/Max	Type	Description
`set_startup_type`	`1..[]`		
`{`			
`name`	`1..1`	`string`	Service name.

startup_type	1..1	string	Startup type: Automatic -- specifies that the service starts automatically when the system starts. Manual -- specifies that a user or a dependent service can start the service. Services with this startup type do not start automatically when the system starts. Disabled -- prevents the service from being started by the system, user, or any dependent service.
}			

Returns:

OK/Error.

Example:

Sets the startup type for `Telnet` service in the specified server to `Manual`.

```
<packet version="4.0.0">
  <dst>
    Host74ee4ead-577a-438d-a22b-978922ecdac0</host>
  </dst>
  <target>servicem</target>
  <data>
    <servicem>
      <set_startup_type>
        <name>TlntSvr</name>
        <startup_type>Manual</startup_type>
      </set_startup_type>
    </servicem>
  </data>
</packet>
```

sessionm

Purpose:

The session management interface. Allows managing client sessions.

Types

sessionType

Summary:

Contains information about a session.

Type specification:

Name	Min/Max	Type	Description
`id`	`1..1`	`string`	The session ID.
`creation`	`1..1`	`datetime_type (p. 25)`	Time of creation.
`access`	`1..1`	`datetime_type (p. 25)`	Last access time.
`user`	`1..1`	`auth_nameType (p. 29)`	The session owner.
`expiration`	`0..1`	`int`	Timeout value. If this element is absent in the Agent response, it means that the session is persistent and never times out.

stamp	0..1	string	*Internal. Not used in client calls.* Session timestamp
data	0..[]	dataType	*Internal. Not used in client calls.* Session-level custom data.
token	0..1	tokenType	The session owner security information.
ip	0..1	ip_type (p. 26)	IP address.

Calls

Call	Description
logout (p. 470)	Logs the user out and destroys a session.
duplicate_session (p. 471)	Creates an additional session for the user.
verify (p. 474)	Verifies the existence and validity of a session.
put (p. 474)	Adds custom data to the session context storage.
get (p. 476)	Retrieves data from the session context storage.
list_sessions (p. 478)	Returns a list of existing sessions.
register_client (p. 480)	Registers a client with the Agent. Used with the count_registered call to implement licensing functionality in the client software.
count_registered (p. 481)	Provides information about the clients that are currently registered with the Agent. Used with the register_client call to implement licensing functionality in the client software.

login_as

Summary:

Logs user in using the specified user SID (security ID).

Request specification:

Name	Min/Max	Type	Description
login_as			
{			
sid	1..1	sidType (p. 27)	User security ID.
}			

Returns:

Session ID or Error.

Name	Min/Max	Type	Description
session_id	0..1	string	The ID of the new session.
token	0..1	tokenType	A token containing the user security information.

Description:

Please see the login call description for more information about the Agent login procedure.

logout

Summary:

Terminates the specified user session.

Request specification:

Name	Min/Max	Type	Description
logout			
{			
session_id	1..1	string	Session ID.
}			

Returns:

OK/Error

Description:

If the session has pending requests, the requests will be canceled. If you have custom data in the session storage, the data will be discarded.

Example:

Input

```
<packet version="4.0.0">
  <target>sessionm</target>
  <data>
    <sessionm>
      <logout>
        <session_id>vzl.40000.c60e1c63-cf1f-467a-ad68-
e9261ac3c22d.10c44685c8btbb3</session_id>
      </logout>
    </sessionm>
  </data>
</packet>
```

Ouput

```
<packet id="dc446862b1t41bb" version="4.0.0">
  <origin>sessionm</origin>
  <session>vzl.30100.c60e1c63-cf1f-467a-ad68-
e9261ac3c22d.ec44685c83t26e9</session>
  <data>
    <sessionm>
      <ok/>
    </sessionm>
  </data>
</packet>
```

duplicate_session

Summary:

Duplicates an existing session.

Request specification:

Name	Min/Max	Type	Description
`duplicate_session`			
`{`			
`session_id`	`0..1`	`string`	The ID of the session to duplicate.

expiration	0..1	int	The session timeout value, in seconds. The specified value will be used for this session only and will not affect the original session. If this element omitted, the default timeout value will be used (defined in the Agent configuration).
}			

Returns:

Name	Min/Max	Type	Description
session_id	0..[]	string	The new session ID.
token	0..1	tokenType	A token containing the user security information.
pass	0..1	auth_nameType (p. 29)	The user login information.

Description:

The `duplicate_session` call creates a new session for the user who is currently associated with another active session (the original session for the same user stays intact). After submitting this request, you may use either session to execute requests in this user's name. Use this call if you need an additional session for the user but don't want to make the user enter his/her login and password again.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>sessionm</target>
  <data>
    <sessionm>
      <duplicate_session>
        <session_id>vzl.40000.4.638a2a56-e689-c340-877d-
bd0470f2448c..dc46721aa5t3f82177r1bfa</session_id>
      </duplicate_session>
    </sessionm>
  </data>
</packet>
```

Ouput

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/sessionm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="26c46725f75t440dr81c" time="2007-06-15T04:59:52+0000" priority="0"
version="4.0.0">
<ns1:origin>sessionm</ns1:origin>
<ns1:target>vzclient4-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:sessionm>
    <ns2:session_id>vzl.40000.4.638a2a56-e689-c340-877d-
bd0470f2448c..fc46721cc8t1c35c305r1bfa</ns2:session_id>
    <ns2:token xsi:type="ns3:tokenType">
      <ns3:user>AQUAAAAAIAFWKopjieZAw4d9vQRw8kSMAAAAAA==</ns3:user>
      <ns3:groups>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMAAAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMAQAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMCgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMAgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMAwAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMBAAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIABWKopjieZAw4d9vQRw8kSMBgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIAFWKopjieZAw4d9vQRw8kSMAAAAAA==</ns3:sid>
      </ns3:groups>
      <ns3:deny_only_sids/>
      <ns3:privileges/>
    </ns2:token>
  </ns2:sessionm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

verify

Summary:

Verifies that the specified session exists and is valid.

Request specification:

Name	Min/Max	Type	Description
verify			
{			
session_id	1..1	string	Session ID.
}			

Returns:

OK/Error

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>sessionm</target>
  <data>
    <sessionm>
      <verify>
        <session_id>vzl.YWRtaW4=.bG9jYWw=.30002.040cbfab-999c-49a9-9985-
27a2a3efc3b7.ac4408afb0t7f818572</session_id>
      </verify>
    </sessionm>
  </data>
</packet>
```

Ouput

```
<packet id="2" version="4.0.0">
  <origin>sessionm</origin>
  <session>vzl.YWRtaW4=.bG9jYWw=.30002.040cbfab-999c-49a9-9985-
27a2a3efc3b7.ac4408afb0t7f818572</session>
  <data>
    <sessionm>
      <ok/>
    </sessionm>
  </data>
</packet>
```

put

Summary:

Adds custom data to the session context storage.

Request specification:

Name	Min/Max	Type	Description
put			
{			

session_id	1..1	string	Session ID.
data	1..[]		User data.
{			
key	1..1	string	Key.
value	0..1	anyType	Value.
}			
}			

Returns:

OK/Error

Description:

The request accepts the supplied key-value pair(s) and adds it to the session context storage. You may choose your own names for the keys and you may store any type of data you require. The data will stay in the storage for as long as the session exists and can be retrieved during that time by using the `get` call. To modify an existing key value, use the key name and a new value. To delete the value, leave the `value` element empty (this does not remove the key from the storage but only changes the value to an empty string). Once the session is destroyed, the data is discarded.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>sessionm</target>
  <data>
    <sessionm>
      <put>
        <session_id>vzl.40000.c60e1c63-cf1f-467a-ad68-
e9261ac3c22d.14c446863b2t7e87</session_id>
        <data>
          <key>mykey</key>
          <value>c29tZSB2YWx1ZQ==</value>
        </data>
      </put>
    </sessionm>
  </data>
</packet>
```

Ouput

```
<packet id="15c44686876t390c" version="4.0.0">
  <origin>sessionm</origin>
  <session>vzl.40000.c60e1c63-cf1f-467a-ad68-
e9261ac3c22d.14c446863b2t7e87</session>
  <data>
    <sessionm>
      <ok/>
    </sessionm>
  </data>
</packet>
```

get

Summary:

Retrieves data from the session context storage.

Request specification:

Name	Min/Max	Type	Description
`get`			
`{`			
`session_id`	`1..1`	`string`	Session ID.

key	0..[]	string	Key.
}			

Returns:

- Empty packet if the specified key does not exist.
- Error on failure.
- Data on success:

Name	Min/Max	Type	Description
data	0..[]		
{			
key	1..1	string	Key.
value	1..1	anyType	Value.
}			

Description:

The request retrieves the data from the session context storage that was saved their earlier using the `put` call (p. 474). Specify the same key name(s) that you used when you were saving the data that you want to retrieve.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>sessionm</target>
  <data>
    <sessionm>
      <get>
       <session_id>vzl.40000.c60e1c63-cf1f-467a-ad68-
e9261ac3c22d.14c446863b2t7e87</session_id>
       <key>mykey</key>
      </get>
    </sessionm>
  </data>
</packet>
```

Ouput

```
<packet id="16c44686934tf3e" version="4.0.0">
  <origin>sessionm</origin>
  <session>vzl.40000.c60e1c63-cf1f-467a-ad68-
e9261ac3c22d.14c446863b2t7e87</session>
  <data>
    <sessionm>
      <data>
        <key>mykey</key>
        <value>c29tZSB2YWx1ZQ==</value>
      </data>
    </sessionm>
  </data>
</packet>
```

list_sessions

Summary:

Returns a list of existing sessions.

Request specification:

Name	Min/Max	Type	Description
`list_sessions`		`none`	

Returns:

Name	Min/Max	Type	Description
`session`		`sessionType`	Session information.

Description:

The call retrieves a list of all existing sessions from the session storage of the local Agent.

Example:

Input

```
<packet version="4.0.0" id="2">
  <target>sessionm</target>
  <data>
    <sessionm>
      <list_sessions/>
    </sessionm>
  </data>
</packet>
```

Ouput

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/sessionm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="27c4672638ft491cr81c" time="2007-06-15T05:11:14+0000" priority="0"
version="4.0.0">
<ns1:origin>sessionm</ns1:origin>
<ns1:target>vzclient4-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:sessionm>
    <ns2:session>
      <ns2:id>vzl.40000.4.638a2a56-e689-c340-877d-
bd0470f2448c..bc46721459t25f803abr1bfa</ns2:id>
      <ns2:user xsi:type="ns3:auth_nameType">
        <ns3:name>cm9vdA==</ns3:name>
        <ns3:realm>00000000-0000-0000-0000-000000000000</ns3:realm>
      </ns2:user>
      <ns2:creation>2007-06-15T04:23:53+0000</ns2:creation>
      <ns2:access>2007-06-15T04:23:53+0000</ns2:access>
      <ns2:expiration>10800</ns2:expiration>
      <ns2:stamp>0.0</ns2:stamp>
    </ns2:session>
    <ns2:session>
      <ns2:id>vzl.40000.65537.638a2a56-e689-c340-877d-
bd0470f2448c..3c46713f60t5082dbdfr1bfa</ns2:id>
      <ns2:user xsi:type="ns3:auth_nameType">
        <ns3:name>dnphZ2VudA==</ns3:name>
        <ns3:realm>00000000-0000-0000-0000-000000000000</ns3:realm>
      </ns2:user>
      <ns2:creation>2007-06-14T13:15:12+0000</ns2:creation>
      <ns2:access>2007-06-14T13:15:12+0000</ns2:access>
      <ns2:stamp>0.0</ns2:stamp>
    </ns2:session>
    <ns2:session>
      <ns2:id>vzl.40000.65537.638a2a56-e689-c340-877d-
bd0470f2448c..7c4671710bt70810fc0r1bfa</ns2:id>
      <ns2:user xsi:type="ns3:auth_nameType">
        <ns3:domain></ns3:domain>
        <ns3:name>cm9vdA==</ns3:name>
        <ns3:realm>00000000-0000-0000-0000-000000000000</ns3:realm>
      </ns2:user>
      <ns2:creation>2007-06-14T16:47:07+0000</ns2:creation>
```

```
      <ns2:access>2007-06-14T16:47:07+0000</ns2:access>
      <ns2:stamp>0.0</ns2:stamp>
    </ns2:session>
  </ns2:sessionm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

register_client

Summary:

Registers a client with the Agent. Used with `count_registered` call (p. 481) to implement licensing functionality in the client software.

Request specification:

Name	Min/Max	Type	Description
`register_client`			
`{`			
`id`	`1..1`	`string`	An arbitrary string representing the client ID. For example, this could be some constant string identifying a version of your client software (to limit the number of simultaneous connections from your software to a given Agent).
`session_id`	`1..1`	`string`	Session ID.
`}`			

Returns:

OK/Error

Description:

The `register_client` call and the `count_registered` call (p. 481) allow to keep track of the logged in clients and to limit the number of concurrent connections from the same client application to a given Agent. The following describes a typical usage scenario.

As soon as a client establishes a connection with Agent, use the `count_registered` call (p. 481) to get the number of currently registered clients with the same ID. Depending on the result, one of the following should happen:

- If the number is less then the maximum allowed number of concurrent connections from the same client (you pick the max number yourself), the login is granted. The new connection is then registered with Agent using the `register_client` call.
- If the number is equals to or greater than the maximum allowed number of connections, the login is denied and the connection is terminated.

It is not necessary to unregister the connection when the client logs off, as Agent does that automatically.

Example:

```
<packet version="4.0.0" id="2">
  <target>sessionm</target>
  <data>
    <sessionm>
      <register_client>
        <id>license_id_333</id>
        <session_id>vzl.40000.65537.1b7066f2-950e-d142-8a56-
dff57c5a305a..5c469223b7t7c5a3d5fr1bb8</session_id>
      </register_client>
    </sessionm>
  </data>
</packet>
```

count_registered

Summary:

Provides information about the clients that are currently registered with the Agent. Used with the `register_client` call (p. 480) to implement licensing functionality in the client software.

Request specification:

Name	Min/Max	Type	Description
`count_registered`			
{			

id	0..1	string	Client ID (see register_client (p. 480)). If the element is omitted, the information for all registered clients will be retrieved.
}			

Returns:

Name	Min/Max	Type	Description
registered	0..1		Client connection information.
{			
id	1..1	string	Client ID.
count	1..1	int	The number of existing connections from this client.
}			

Description:

The `count_registered` call and the `register_client` call (p. 480) are used together. They allow to keep track of logged in clients and to limit the number of concurrent connections from the same client by granting or denying a new connection based on the number of connections that already exist. The following describes a typical usage scenario.

As soon as a client establishes a connection with PVA Agent, use the `count_registered` call to get the number of currently registered clients with the same ID. Depending on the result, one of the following should happen:

- If the number is less then the maximum allowed concurrent connections (the maximum number is determined by you) the login is granted. The new connection is then registered with Agent using the `register_client` call (p. 480).
- If the number is equals to or greater than the maximum allowed connections, the login is denied and the connection is terminated.

It is not necessary to unregister the connection when the client logs off, as Agent does that automatically.

Example:

Input

```
<packet version="4.0.0">
  <target>sessionm</target>
  <data>
    <sessionm>
      <count_registered/>
    </sessionm>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/sessionm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="1ac46926d8ft7e87r3c4" time="2007-07-09T17:35:50+0000" priority="0"
version="4.0.0">
<ns1:origin>sessionm</ns1:origin>
<ns1:target>vzclient3-1b7066f2-950e-d142-8a56-dff57c5a305a</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:sessionm>
    <ns2:registered>
      <ns2:id>license_id_333</ns2:id>
      <ns2:count>2</ns2:count>
    </ns2:registered>
  </ns2:sessionm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

userm

Purpose:

The user and group management interface.

Calls

Call	Description
add_user (p. 484)	Adds a new user to a server.
del_user (p. 487)	Deletes a user from a server.
edit_user (p. 488)	Updates the user information.
del_group (p. 489)	Deletes a group from a server.
add_group (p. 490)	Adds a new group to a server.
get_user (p. 492)	Retrieves the user information.
group_add_user (p. 494)	Adds a user to a group.
group_del_user (p. 495)	Removes a user from a group.
get_group (p. 496)	Retrieves the group information.
edit_group (p. 498)	Allows to modify name and GID for an existing group.
group_set_users (p. 499)	Removes all existing users from a group and adds the specified users.
get_limits (p. 500)	Retrieves the server uid/gid limits.
authenticate (p. 502)	Authenticates the specified user.

add_user

Summary:

Adds a new user to a server.

Request specification:

Name	Min/Max	Type	Description
`add_user`	`0..[]`		
`{`			
`  user`	`1..1`	`userType (p. 61)`	The new user information.

min_uid	0..1	int	Lower bound of the UID (user ID) range. This element, together with the max_uid element (below), specifies the UID range. If there is an unused ID in the range then it will be used for the new user.
max_uid	0..1	int	Upper bound of the UID range.
create_home_dir	0..1	boolean	Specifies whether a home directory should be created for the user. Possible values: true -- Create home directory. false -- Do not create the directory. If the parameter is omitted, the decision whether to create a directory or not will be made automatically based on the OS template used (some OS templates imply the creation of home directories while others don't). If the parameter is included, the default template behaviour is ignored.
create_initial_group	0..1		Specifies whether to create an initial group for the user. In order to create the initial group, include this element and the user/initial_group/name element containing the name of group. Some Linux distributions will create the initial group with the same name as the user name by default. Other distributions will add a new user to a predefined group. You can use this field, together with the user/initial_group/name field, to override this default behaviour.
}			

Returns:

Name	Min/Max	Type	Description
user	0..[]	userType	The new user information.

Example:

Adding a new user named "TestUser" to the specified server. The Server ID is specified using the `dst` element of the packet header. Also creating an initial group named "InitGroup" and adding the user to it. The user ID will be selected from the range 100-200.

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <add_user>
        <user>
          <initial_group>
            <name>InitGroup</name>
          </initial_group>
          <name>TestUser</name>
          <shell>/bin/bash</shell>
        </user>
        <min_uid>100</min_uid>
        <max_uid>200</max_uid>
        <create_initial_group/>
      </add_user>
    </userm>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/userm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="1dc45eee46at4db7rdfc" time="2007-03-07T11:53:52+0000" priority="0"
version="4.0.0">
<ns1:origin>9bafbeb7-85f7-499e-a210-40e00850a5f3</ns1:origin>
<ns1:dst>
  <director>gend</director>
  <target>vzclient3</target>
</ns1:dst>
<ns1:data>
  <ns2:userm>
    <ns2:user xsi:type="ns3:userType">
      <ns3:uid>101</ns3:uid>
      <ns3:name>TestUser</ns3:name>
      <ns3:initial_group>
        <ns3:gid>500</ns3:gid>
        <ns3:name>InitGroup</ns3:name>
      </ns3:initial_group>
      <ns3:shell>/bin/bash</ns3:shell>
      <ns3:home_dir>/home/TestUser</ns3:home_dir>
    </ns2:user>
  </ns2:userm>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>userm</ns1:target>
</ns1:src>
<ns1:target>vzclient3</ns1:target>
</ns1:packet>
```

del_user

Summary:

Deletes an existing user from a server.

Request specification:

Name	Min/Max	Type	Description
del_user	0..[]		
{			
user	1..1		The user to delete.
{			
name	1..1	string	User name.
}			
remove_home_dir	0..1	none	Include this element if you would also like to remove the user's home directory. The directory must exist, otherwise you will get an error.
remove_initial_group	0..1	none	Include this element if you would also like to remove the user's initial group. The group must not contain any other users, otherwise the call will fail.
}			

Returns:

OK/Error

Example:

Deleting the user TestUser from the specified server.

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <del_user>
        <user>
          <name>TestUser</name>
        </user>
      </del_user>
    </userm>
  </data>
</packet>
```

edit_user

Summary:

Modifies an existing user.

Request specification:

Name	Min/Max	Type	Description
edit_user	0..[]		
{			
name	1..1	string	The name of the user to update the details for.
user	1..1	userType (p. 61)	User information. The user name must always be included, even if you are not changing it.
min_uid	0..1	int	Lower bound of the UID (user ID) range. Include this element if you would like to change the UID. This element, together with the max_uid element (below), specifies the UID range. If there is an unused ID in the range then it will be assigned to the user.
max_uid	0..1	int	Upper bound of the UID range.
}			

Returns:

OK/Error

Example 1:

Changing the user's password and the default shell information in the specified server.

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <edit_user>
        <name>TestUser</name>
        <user>
          <shell>/bin/sh</shell>
          <password>bXluZXdwYXNz</password>
          <name>TestUser</name>
        </user>
      </edit_user>
    </userm>
  </data>
</packet>
```

Example 2:

Changing the user name and adding the user to the specified group.

Input

```
<packet version="4.0.0">
  <target>userm</target>
  <data>
    <userm>
      <edit_user>
        <name>TestUser</name>
        <user>
          <name>JohnDoe</name>
          <group>
            <name>Users</name>
          </group>
        </user>
      </edit_user>
    </userm>
  </data>
</packet>
```

del_group

Summary:

Deletes a group from a server.

Request specification:

Name	Min/Max	Type	Description
del_group	0..[]		

{			
group	1..1		Group to delete.
{			
name	0..1	string	Group name.
}			
}			

Returns:

OK/Errors

Description:

The group must not contain any users, otherwise the call will fail. To delete a non-empty group, remove all users from the group first (you only have to remove the users from the group, you don't have to delete the users from the system).

add_group

Summary:

Adds a group to a server.

Request specification:

Name	Min/Max	Type	Description
add_group	0..[]		
{			
group	1..1		Group to add.
{			
name	1..1	string	Group name.
}			
min_gid	0..1	int	Lower bound of the GID (group ID) range. This element, together with the max_gid element (below), specifies the GID range. If there is an unused ID in the range then it will be used for the new group.
max_gid	0..1	int	Upper bound of the GID range.
}			

Returns:

Name	Min/Max	Type	Description
group	0..[]	groupType	The new group information.

Example:

Creating a new group named "TestGroup" in the specified server. The group ID will be selected from the specified GID range.

Input

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <add_group>
        <group>
          <name>TestGroup</name>
        </group>
        <min_gid>100</min_gid>
        <max_gid>200</max_gid>
        <create_initial_group/>
      </add_group>
    </userm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/userm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="2c46012676t18berf08" time="2007-03-16T22:12:46+0000" priority="0"
version="4.0.0">
<ns1:origin>9bafbeb7-85f7-499e-a210-40e00850a5f3</ns1:origin>
<ns1:dst>
  <director>gend</director>
  <target>vzclient3</target>
</ns1:dst>
<ns1:data>
  <ns2:userm>
    <ns2:group xsi:type="ns3:groupType">
      <ns3:gid>102</ns3:gid>
      <ns3:name>TestGroup</ns3:name>
    </ns2:group>
  </ns2:userm>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>userm</ns1:target>
</ns1:src>
<ns1:target>vzclient3</ns1:target>
</ns1:packet>
```

get_user

Summary:

Retrieves the specified user information.

Request specification:

Name	Min/Max	Type	Description
`get_user`	`0..[]`		
`{`			
`  user`	`0..1`		The user to get the info for. If this element is omitted, the information about all existing users will be retrieved.
`  {`			
`    uid`	`0..1`	`int`	User ID.
`    name`	`0..1`	`string`	User name.
`  }`			
`}`			

Returns:

Name	Min/Max	Type	Description
`user`	`0..[]`	`userType`	User information.

Example:

Retrieving the user `root` details from the specified server.

Input:

```
<packet version="4.0.0" id="2">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <get_user>
        <user>
          <name>root</name>
        </user>
      </get_user>
    </userm>
  </data>
</packet>
```

Output:

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/userm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="4c46012874t4ae1rf08" time="2007-03-16T22:18:22+0000" priority="0"
version="4.0.0">
<ns1:origin>9bafbeb7-85f7-499e-a210-40e00850a5f3</ns1:origin>
<ns1:dst>
  <director>gend</director>
  <target>vzclient3</target>
</ns1:dst>
<ns1:data>
  <ns2:userm>
    <ns2:user xsi:type="ns3:userType">
      <ns3:uid>0</ns3:uid>
      <ns3:name>root</ns3:name>
      <ns3:initial_group>
        <ns3:gid>0</ns3:gid>
        <ns3:name>root</ns3:name>
      </ns3:initial_group>
      <ns3:group>
        <ns3:gid>0</ns3:gid>
        <ns3:name>root</ns3:name>
      </ns3:group>
      <ns3:group>
        <ns3:gid>1</ns3:gid>
        <ns3:name>bin</ns3:name>
      </ns3:group>
      <ns3:group>
        <ns3:gid>2</ns3:gid>
        <ns3:name>daemon</ns3:name>
      </ns3:group>
      <ns3:group>
        <ns3:gid>3</ns3:gid>
        <ns3:name>sys</ns3:name>
      </ns3:group>
      <ns3:group>
        <ns3:gid>4</ns3:gid>
        <ns3:name>adm</ns3:name>
      </ns3:group>
      <ns3:group>
```

```
        <ns3:gid>6</ns3:gid>
        <ns3:name>disk</ns3:name>
      </ns3:group>
      <ns3:group>
        <ns3:gid>10</ns3:gid>
        <ns3:name>wheel</ns3:name>
      </ns3:group>
      <ns3:shell>/bin/bash</ns3:shell>
      <ns3:home_dir>/root</ns3:home_dir>
      <ns3:comment>root</ns3:comment>
    </ns2:user>
  </ns2:userm>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>userm</ns1:target>
</ns1:src>
<ns1:target>vzclient3</ns1:target>
</ns1:packet>
```

group_add_user

Summary:

Adds a user or a member group to a group.

Request specification:

Name	Min/Max	Type	Description
group_add_user	0..[]		
{			
group	1..1		Group to add the user to (the group must already exist).
{			
name	0..1	string	The group name.
}			
[			
user	1..1		The user to add to the group (the user must already exist).
{			
name	0..1	string	User name.
}			
member_group	1..1		The member group to add to the group (the group that you are adding as a member group must already exist).
{			
name	0..1	string	Group name.
}			

]			
}			

Returns:

OK/Error

Example:

Adding the user `TestUser` to the group `TestGroup`.

```
<packet version="4.0.0" id="2">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <group_add_user>
        <group>
          <name>TestGroup</name>
        </group>
        <user>
          <name>TestUser</name>
        </user>
      </group_add_user>
    </userm>
  </data>
</packet>
```

group_del_user

Summary:

Removes a user or a member group from a group.

Request specification:

Name	Min/Max	Type	Description
group_del_user	0..[]		
{			
group	1..1		Parent group information.
{			
name	1..1	string	The group name.
}			
[			
user	1..1		The user to remove from the parent group.
{			
name	0..1	string	User name.
}			
member_group	0..1		The member group to remove from the parent group.

{			
name	0..1	string	Group name.
}			
]			
}			

Returns:

OK/Errors

Description:

The call removes a user or a member group from a parent group. The call does not delete a user or a group from the system, it simply cancels their membership in the parent group.

Example:

Removing the user `TestUser` from the group `TestGroup` in the specified server.

```
<packet version="4.0.0" id="2">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <group_del_user>
        <group>
          <name>TestGroup</name>
        </group>
        <user>
          <name>TestUser</name>
        </user>
      </group_del_user>
    </userm>
  </data>
</packet>
```

get_group

Summary:

Retrieves the specified group information.

Request specification:

Name	Min/Max	Type	Description
get_group	0..[]		
{			
group	0..1		The group to get the info for.
{			
name	0..1	string	Group name.
gid	0..1	string	Group ID.

}			
}			

Returns:

Name	Min/Max	Type	Description
group	0..[]	groupType	Group information.

Example:

Retrieving the group TestGroup details from the specified server.

Input

```
<packet version="4.0.0" id="2">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <get_group>
        <group>
          <name>TestGroup</name>
        </group>
      </get_group>
    </userm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/userm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="7c46012dddt72aerf08" time="2007-03-16T22:34:02+0000" priority="0"
version="4.0.0">
<ns1:origin>9bafbeb7-85f7-499e-a210-40e00850a5f3</ns1:origin>
<ns1:dst>
  <ns1:director>gend</ns1:director>
  <ns1:target>vzclient3</ns1:target>
</ns1:dst>
<ns1:data>
  <ns2:userm>
    <ns2:group xsi:type="ns3:groupType">
      <ns3:gid>102</ns3:gid>
      <ns3:name>TestGroup</ns3:name>
    </ns2:group>
  </ns2:userm>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>userm</ns1:target>
</ns1:src>
<ns1:target>vzclient3</ns1:target>
</ns1:packet>
```

edit_group

Summary:

Allows to modify name and GID for an existing group.

Request specification:

Name	Min/Max	Type	Description
edit_group	0..[]		
{			
name	1..1	string	The group to update the info for.
group	1..1		New name and (or) GID.
{			
name	0..1	string	New group name.
gid	0..1	int	New group ID.
}			
min_gid	0..1	int	Lower bound of the GID (group ID) range. Include this element if you would like to change the group GID and would like it to be selected automatically from the specified range. This element, together with the max_gid element (below), specifies the GID range. If there is an unused ID in the range then it will be used for the new group.
max_gid	0..1	int	Upper bound of the GID (group ID) range.
}			

Returns:

OK/Error

Example:

Changing the name of an existing group `TestGroup` to `mynewgroup`. Also changing the group ID -- the ID will be automatically selected from the range of 300 to 400.

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <edit_group>
        <name>TestGroup</name>
        <group>
          <name>mynewgroup</name>
        </group>
        <min_gid>300</min_gid>
        <max_gid>400</max_gid>
      </edit_group>
    </userm>
  </data>
</packet>
```

group_set_users

Summary:

Removes all members from the specified group and adds the specified members.

Request specification:

Name	Min/Max	Type	Description
group_set_users	0..[]		
{			
group	1..1		Parent group information.
{			
name	1..1	string	The group name.
}			
user	0..[]		Users to add to the parent group.
{			
name	1..1	string	User name.
}			
member_group	0..[]		Member groups to add to the parent group.
{			
name	0..1	string	Group name.

}			
}			

Returns:

OK/Error

Description:

The call will first remove all users and member groups from the specified group (the users and the groups will not be deleted from the system). It will then add the specified users and (or) member groups to the parent group. To simply remove all members from a group without adding the new ones, do not include the `user` and the `member_group` parameters.

Example:

Removing the user `TestUser` from the group `mynewgroup`.

```
<packet version="4.0.0">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <group_set_users>
        <group>
          <name>mynewgroup</name>
        </group>
        <user>
          <name>TestUser</name>
        </user>
      </group_set_users>
    </userm>
  </data>
</packet>
```

get_limits

This call is available on Linux only.

Summary:

Gets boundary values for min/max user and group IDs for the specified server.

Request specification:

Name	Min/Max	Type	Description
`get_limits`	`0..[]`		

Returns:

Name	Min/Max	Type	Description
`limits`	`0..1`		Limits for user and group creation.
{			

umin	1..1	int	min UID used for user creation.
umax	1..1	int	max UID used for user creation.
gmin	1..1	int	min GID used for group creation.
gmax	1..1	int	max GID used for group creation.
}			

Example:

Input

```
<packet version="4.0.0" id="2">
  <dst>
    Host9bafbeb7-85f7-499e-a210-40e00850a5f3</host>
  </dst>
  <target>userm</target>
  <data>
    <userm>
      <get_limits/>
    </userm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/userm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="0"
id="12c460136b1t12dbrf08" time="2007-03-16T22:59:40+0000" priority="0"
version="4.0.0">
<ns1:origin>9bafbeb7-85f7-499e-a210-40e00850a5f3</ns1:origin>
<ns1:dst>
  <director>gend</director>
  <target>vzclient3</target>
</ns1:dst>
<ns1:data>
  <ns2:userm>
    <ns2:limits>
      <ns2:umin>500</ns2:umin>
      <ns2:umax>60000</ns2:umax>
      <ns2:gmin>500</ns2:gmin>
      <ns2:gmax>60000</ns2:gmax>
    </ns2:limits>
  </ns2:userm>
</ns1:data>
<ns1:src>
  <ns1:director>vpsd</ns1:director>
  <ns1:target>userm</ns1:target>
</ns1:src>
<ns1:target>vzclient3</ns1:target>
</ns1:packet>
```

authenticate

Summary:

Verify user authenticity for the specified username and password.

Request specification:

Name	Min/Max	Type	Description
`authenticate`	`0..1`		
{			
`name`	`1..1`	`string`	User name.
`password`	`1..1`	`base64Binary`	Password.
`type`	`0..1`	`string`	Authentication type. Possible values: `system` (default) -- local system account. `system_admin` -- system administrator. `system_admin_group` -- a user belonging to the System Administrators group.
}			

Returns:

Name	Min/Max	Type	Description
`token`	`0..1`	`tokenType`	Authentication results.

Example:

Authenticating the user `root`.

Input

```
<packet version="4.0.0" id="2">
  <target>userm</target>
  <data>
    <userm>
      <authenticate>
         <name>root</name>
         <password>bXlwYXNz</password>
      </authenticate>
    </userm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/userm"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="16c4601384etf3erf08"
time="2007-03-16T23:04:10+0000" priority="0" version="4.0.0">
<ns1:origin>userm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns2:userm>
    <ns2:token xsi:type="ns3:tokenType">
      <ns3:user>AQUAAAAAA+hWW5a9ohIJQ6S9aykBIAAAAAAAAA==</ns3:user>
      <ns3:groups>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAAAAAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAAAQAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAACgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAAAgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAAAwAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAABAAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykAIAAABgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAA+hWW5a9ohIJQ6S9aykBIAAAAAAAAA==</ns3:sid>
      </ns3:groups>
      <ns3:deny_only_sids/>
      <ns3:privileges/>
    </ns2:token>
  </ns2:userm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Events

System events are produced by Agent and are used to inform the client of a change in the system operations. Such events as a Container status change, configuration change, and resource allocation alerts are currently supported. Other event types may be added in the future.

Agent monitors the system at all times and triggers an event as soon as it detects the corresponding change. In order to be automatically notified of an event, the client must subscribe to the event notification services first. This is accomplished by executing the `subscribe` (p. 558) or `subscribe_alert` (p. 68) calls.

The client subscribes to a particular event type by specifying the *event subscription name*, which is defined for every event type individually. The Elements section (p. 506) describes the event types and their corresponding subscription names. The subscription name is used when you subscribe to an event. The same name is then used as the value of the `target` element of the event notification message, so that you can recognize the event message among other messages that you might be receiving from the Agent server side.

The following subsections describe currently supported event types and provide examples.

Types

env_status_event_dataType

Summary:

Contains information about the server status change.

Type definition:

Extends `event_dataType (p. 35)`

Adds the following elements:

Name	Min/Max	Type	Description
`eid`	`1..1`	`eid_type (p. 25)`	The ID of the affected server.
`parent_eid`	`1..1`	`eid_type (p. 25)`	Parent server ID.
`new`	`1..1`	`env_statusType (p. 34)`	New status/transition.
`old`	`1..1`	`env_statusType (p. 34)`	Old status/transition.

Event category:

env_status

env_config_event_dataType

Summary:

Contains information about the server configuration change.

Type definition:

Extends event_dataType (p. 35)

Adds the following elements:

Name	Min/Max	Type	Description
config	0..1	env_configType	The server configuration data.
virtual_config	0..1	env_configType	Virtual configuration.
eid	1..1	eid_type (p. 25)	The ID of the affected server.
parent_eid	1..1	eid_type (p. 25)	Parent server ID.

Event category:

env_config

resource_alertType

Summary:

Contains information about an alert that was raised on a server.

Type definition:

Extends alert_dataType (p. 28)

Adds the following elements:

Name	Min/Max	Type	Description
type	1..1	int	Alert level. See alert_dataType (p. 28) for more info.
eid	0..1	eid_type (p. 25)	Server ID.

class	1..1	string	The class of the performance counter (see Parallels Agent Programmer's Guide for more information on performance counters, classes, and instances).
instance	1..1	string	Performance class instance.
counter	1..1	string	Performance counter.
cur	1..1	xs:anySimpleType	Current value.
hard	1..1	xs:anySimpleType	Hard limit value.
soft	1..1	xs:anySimpleType	Soft limit value.

Event category:

resource_alert

Elements

When an XML packet containing the event notification message is received by the client program, the event data is contained in one of the elements described in this section. Each element corresponds to a particular event type.

env_status_event

Purpose:

Notifies about changes in the server status, including state and/or transition changes.

Event specification:

Name	Min/Max	Type
env_status_event	1..1	env_status_event_dataType (p. 504)

Subscription name:

`env_status_subscription`

See `subscribe` (p. 558) for more information about subscriptions.

Description:

This event triggers when the status of a server changes (including state and transition changes). The event reports the status changes for every server that the Agent is aware of. If you subscribe to the event on the Master Node in a Virtuozzo group, you will receive notifications about the status changes for every server (physical or virtual) in the entire group.

The `env_status_event` element substitutes the `event_data` element in the `eventType` structure (p. 36).

Example:

Input

Subscribing to the environment status change events.

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <subscribe>
        <name>env_status_subscription</name>
      </subscribe>
    </system>
  </data>
</packet>
```

Output

Subscription was a success.

```
<packet priority="0" version="4.0.0" id="2">
  <origin>servd</origin>
  <target>vzclient1</target>
  <data>
    <system>
      <ok/>
    </system>
  </data>
</packet>
```

Output

The following is a notification message that was received when one of the Environments was manually stopped. The message contains the environment ID that generated the event, the text message describing the event, and the event data (old/new environment state and transition codes). Note that one of the `target` elements contains the same value as the one we used in the `name` element in the request, which is `env_status_subscription`. Please also note that the inner `data` structure contains the elements specific to this event type. In this particular case, this is the `env_status_event` element.

```
<packet version="4.0.0">
  <target>events_subscription</target>
```

```
  <target>env_status_subscription</target>
  <data>
    <event>
      <eid>849c9be9-5fbb-4e7d-b100-f841f86c150e</eid>
      <time>1155317636</time>
      <source></source>
      <category>env_status_subscription</category>
      <sid>XXX</sid>
      <data>
        <env_status_event>
          <eid>62ec514e-bc38-4aee-830d-cc802ee2aadd</eid>
          <new>
            <state>3</state>
          </new>
          <old>
            <state>3</state>
            <transition>5</transition>
          </old>
        </env_status_event>
      </data>
      <info>
        <message>

RW52aXJvbm1lbnQgJWVpZCUgc3RhdHVzIGNoYW5nZWQgZnJvbSAlb2xkJSB0byAlbmV3JQ==
        </message>
        <name></name>
        <translate/>
        <parameter>
          <message>NjJlYzUxNGUtYmMzOC00YWVlLTgzMGQtY2M4MDJlZTJhYWRk</message>
          <name>eid</name>
        </parameter>
        <parameter>
          <message>Mw==</message>
          <name>new</name>
          <translate/>
        </parameter>
        <parameter>
          <message>Mw==</message>
          <name>old</name>
          <translate/>
        </parameter>
      </info>
    </event>
  </data>
</packet>
```

env_config_event

Purpose:

The event reporter that notifies about changes in the environment configuration.

Event specification:

Substitution group: `event_data` **(p. 36)**

Name	Min/Max	Type
`env_config_event`	`1..1`	`env_config_event_dataType` (p. 505)

Subscription name:

`env_config_subscription`

See `subscribe` (p. 558) for more information about subscriptions.

Description:

This event triggers when the configuration of a server changes. The event reports configuration changes for every server that the Agent is aware of. If you subscribe for the event on the Master Node in a Virtuozzo group, you will receive the notifications about the configuration changes of every server in the entire group.

The `env_config_event` element substitutes the `event_data` element in the `eventType` structure **(p. 36)**.

Example:

Input

Subscribing to the environment configuration change events.

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <subscribe>
        <name>env_config_subscription</name>
      </subscribe>
    </system>
  </data>
</packet>
```

Output

Subscription was a success.

```
<packet id="2" priority="0" version="4.0.0">
  <origin>gend</origin>
  <target>vzclient2</target>
  <dst>
    <director>gend</director>
  </dst>
  <data>
    <system>
      <ok/>
    </system>
  </data>
</packet>
```

Output

The following is a notification message received when one of the configuration of one of the Environments was manually changed. The message contains the environment ID that generated the event, the text message that may be presented to the user, and the event data (the new configuration information). Note that one of the `target` elements contains the same value as the one we used in the `name` element of the request, which is `env_config_subscription`. Please also note that the inner `data` structure contains the elements specific to this event type. In this particular case, this is the `env_config_event` element.

```
<packet version="4.0.0" time="2006-08-12T08:53:16+0000">
  <target>events_subscription</target>
  <target>env_config_subscription</target>
  <src>
    <director>gend</director>
  </src>
  <data>
    <event>
      <eid>62ec514e-bc38-4aee-830d-cc802ee2aadd</eid>
      <time>1155372796</time>
      <source></source>
      <category>env_config_subscription</category>
      <sid>XXX</sid>
      <data>
        <env_config_event>
          <eid>62ec514e-bc38-4aee-830d-cc802ee2aadd</eid>
          <virtual_config>
            <offline_management>1</offline_management>
            <on_boot>0</on_boot>
            <os_template>
              <version>20060615</version>
              <name>redhat-as3-minimal</name>
            </os_template>
            <ve_root>/vz/root/$VEID</ve_root>
            <ve_private>/vz/private/$VEID</ve_private>
            <address>
              <ip>10.17.4.132</ip>
            </address>
            <address>
              <ip>10.17.5.132</ip>
            </address>
            <hostname>myhost1</hostname>
            <qos>
              <id>avnumproc</id>
              <hard>40</hard>
            </qos>
            <qos>
              <id>cpuunits</id>
              <hard>1000</hard>
            </qos>
            <qos>
              <id>dcachesize</id>
              <hard>1097728</hard>
              <soft>1048576</soft>
            </qos>
            <qos>
              <id>dgramrcvbuf</id>
              <hard>132096</hard>
              <soft>132096</soft>
            </qos>

            <!-- the rest of the QoSs are omitted for brevity -->
            <veid>101</veid>
            <type>virtuozzo</type>
            <disabled>0</disabled>
            Linux
              <platform/>
              <version>20060615</version>
              <name>redhat-as3-minimal</name>
            </os>
          </virtual_config>
        </env_config_event>
      </data>
      <info>
        <message>RW52aXJvbWVudCAlZWlkJSBjb25maWcgY2hhbmdlZA==</message>
        <name></name>
        <translate/>
```

```
        <parameter>
          <message>NjJlYzUxNGUtYmMzOC00YWVlLTgzMGQtY2M4MDJlZTJhYWRk</message>
          <name>eid</name>
        </parameter>
      </info>
    </event>
  </data>
</packet>
```

resource_alert

Purpose:

Reports the resource allocation problems such as approaching or exceeding certain limits.

Event specification:

Name	Min/Max	Type
`resource_alert`	`1..1`	`resource_alertType` (p. 505)

Subscription name:

`alerts_subscription`

See `subscribe` (p. 558) for more information about subscriptions.

Description:

This event triggers when an alert is raised on a server. The event reports alerts for every server that the Agent is aware of. If you subscribe to the event on the Master Node in a Virtuozzo group, you will receive the alert notifications about every server in the entire group.

The `resource_alert` element substitutes the `event_data` element in the `eventType` structure **(p. 36)**.

Some of the alerts deal with the resource allocations. Resource alert is a notification about some parameter of the system, getting over some barrier, or coming close to some limit. Usually they are used for monitoring of the environment health, predicting its performance, or collecting information about the settings in need of tuning.

There are four possible alert levels:

Alert level	ID	Description
Green	0	Everything is fine. This alert is raised when one of the higher-level alerts is canceled.
Yellow	1	Moderately dangerous situation. The specified parameter is coming close (within 10%) to its soft limit barrier.
Red	2	Critical situation. The parameter exceeded its soft limit or came very close to the hard limit. Depending on the parameter type, either some process can be killed at any time now, or the next resource allocation request can be refused.

It is always a good idea to check and adjust the QoS configuration parameters upon receiving an alert.

Example:

Input

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <subscribe>
        <name>alerts_subscription</name>
      </subscribe>
    </system>
  </data>
</packet>
```

Output:

```
<packet version="4.0.0" time="2006-08-18T10:47:43+0000">
  <target>events_subscription</target>
  <target>alerts_subscription</target>
  <src>
    <director>gend</director>
  </src>
  <data>
    <event>
      <eid>ccc794ad-cc5d-49f2-8d84-6631263c81be</eid>
      <time>2006-08-18T10:47:43+0000</time>
      <source>resource_alert_monitor</source>
      <category>resource_alert</category>
      <sid>XXX</sid>
      <data>
        <resource_alert>
          <eid>13a10bc2-3ace-4bf9-ac0f-e33d98b1766d</eid>
          <type>1</type>
          <class>counters_vz_ubc</class>
          <counter>numproc</counter>
          <cur>8</cur>
          <hard>10</hard>
        </resource_alert>
      </data>
      <info>
        <message>
UmVzb3VyY2UgJXR5cGUlICVpZCUgYWxlcnQgb24gZW52aXJvbm1lbnQgJWVpZCUgY3VycmVudCB2YW
x1ZTogJWN1ciUgc29mdCBsaW1pdDogJXNvZnQlIGhhcmQgbGltaXQ6ICVoYXJkJQ==
        </message>
        <name></name>
        <translate/>
          <parameter>
            <message>OA==</message>
            <name>cur</name>
          </parameter>
          <parameter>

<message>MTNhMTBiYzItM2FjZS00YmY5LWFjMGYtZTMzZDk4YjE3NjZk</message>
            <name>eid</name>
          </parameter>
          <parameter>
            <message>MTA=</message>
            <name>hard</name>
          </parameter>
          <parameter>
            <message>bnVtcHJvYw==</message>
            <name>id</name>
          </parameter>
```

```
          <parameter>
            <message></message>
            <name>soft</name>
          </parameter>
          <parameter>
            <message>eWVsbG93</message>
            <name>type</name>
          <translate/>
        </parameter>
      </info>
    </event>
  </data>
</packet>
```

System Interface and Special Packets

system

Purpose:

The `system` interface provides calls for controlling the Agent processing of client requests, getting the system-level Agent information, and for performing miscellaneous Agent system tasks. Compared to the rest of the Agent XML API calls, the `system` interface calls are used slightly differently and have certain limitations as described below:

- Unlike most of the other Agent calls, the `system` interface calls do not have a target operator. This means that when you are composing XML messages from these specifications, you do not include the `target` element in the message header (see code examples at the end of each section describing a call).
- An individual XML packet sent to the Agent cannot have more than one `system` request. In addition, a request cannot have more than one `system` interface call. This means that you cannot have more than one `system` element in a packet and more than one call under that element. If you want to execute multiple `system` calls in succession, you will have to prepare and send a separate complete XML packet for each call.

Types

voc_parameterType

Summary:

Contains an Agent vocabulary entry information.

Type specification:

Name	Min/Max	Type	Description
`id`	`1..1`	`string`	Entry ID.
`type`	`0..1`	`string`	Data type (int, string, etc.)

min	0..1	string	Minimum possible value.
max	0..1	string	Maximum possible value.
long	0..1	string	Long description.
short	0..1	string	Short description.
category	0..[]	string	Vocabulary category.
complex	0..1	string	The entry type-specific value. Used when the parameter data type has a complex structure. The values contained in this element may have different meanings for different types of vocabulary entries.
default	0..1	string	The default value.
measure	0..1	string	Units of measure.
data	0..1		Data. Contains an entry-type specific value.

Calls

Call	Description
subscribe (p. 558)	Subscribes for an event notification service.
unsubscribe (p. 562)	Cancels an event subscription.
cancel (p. 516)	Cancels processing of the specified Agent request.
get_state (p. 547)	Gets the state of the Agent director.
get_configuration (p. 533)	Returns current Agent configuration.
get_version (p. 550)	Returns current Agent version information. This is NOT the protocol version (i.e. 4.0.0) but the internal Agent version code.
get_realm (p. 545)	Retrieves the list of the available Realms from the Agent configuration.
register_client (p. 557)	Registers a client with the Agent. Used with count_registered call (p. 527) to implement licensing functionality in the client software.
count_registered (p. 527)	Provides information about the clients that are currently registered with the Agent. Used with the register_client call (p. 557) to implement licensing functionality in the client software.
get_vocabulary (p. 551)	Retrieves the Agent vocabulary data.
ping (p. 555)	A simple ping function. Used to to test the availability of a host on a network.
connect (p. 525)	Establishes a new exclusive connection with a remote Agent.

close (p. 517)	Closes the exclusive connection, previously opened by the `connect` call (p. 525).
distribute (p. 530)	Distributes Agent core to another node.
uninstall (p. 561)	Uninstalls Agent from a server.
get_plugins (p. 541)	Retrieves the list of the installed Agent plug-in modules.
get_eid (p. 541)	Gets the Server ID of the computer you are currently connected to.
login (p. 553)	Opens a permanent connection to the Agent and logs the user in.
generate_pass (p. 532)	Generates a one-time login and password that can be used to connect to the Agent on the specified server.
configuration (p. 518)	Allows to modify Agent configuration.

cancel

Summary:

Cancels the processing of a client request.

Request specification:

Name	Min/Max	Type	Description
`cancel`			
`{`			
`message_id`	`1..1`	`string`	Request ID. You must assign this ID manually to every message for which you plan on providing this functionality in your client applications. The ID is assigned using the `id` attribute of the `packet` element when you compose the request.
`target`	`0..1`	`string`	The target operator that was specified in the original request. The message ID that exists within a particular Agent session may not be unique within the entire Agent server context. Use this parameter together with `message_id` (above) to make sure you are canceling the correct request.
`global`	`0..1`	`none`	Normally, Agent will verify that the request that you are trying to cancel and this request both originate on the same client. Include this element to bypass this verification. For this to succeed, the message ID specified in the `message_id` parameter must be unique.
`}`			

Returns:

OK/Error

Description:

Use the `cancel` call to cancel a client request which is currently being processed by Agent. Whether the request can be canceled or not depends on how long it takes to process the request and the number of stages the entire request processing is divided into. To cancel a request, you have to know its ID in advance. The ID is assigned to a request manually at the time it is composed.

close

Summary:

Closes an exclusive Agent connection that was previously opened using the `connect` call (p. 525).

Request specification:

Name	Min/Max	Type	Description
`close`			
`{`			
`id`	`1..1`	`string`	The ID of the exclusive connection to close. The ID is initially obtained from the return of the `connect` call (p. 525) that you used to established the connection.
`}`			

Returns:

OK/Error

Example:

```
<packet version="4.0.0">
  <data>
    <system>
      <close>
        <id>192.168.0.844</id>
      </close>
    </system>
  </data>
</packet>
```

configuration

Summary:

Sets the Agent configuration.

Request specification:

Name	Min/Max	Type	Description
`configuration`	`0..1`	`none`	Agent configuration parameters. Use the `get_configuration` call (p. 533) to retrieve the existing configuration, modify the values as needed, and then use the entire structure as an input here.

Returns:

The OK or Error message for every Agent operator the configuration of which has been changed.

Description:

Normally, the majority of Agent configuration parameters should never be modified. The only parameters that you might be interested in changing is the timeout values for Agent operators, and some miscellaneous parameters such as `mute_alert_period` as an example (see alert subscriptions (p. 68)).

Example:

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="45c466fe763t767dra18" priority="0" version="4.0.0">
<data>
  <system>
    <configuration>
      <distribution>
        <item>
          <path>/opt/vzagent/distribution/distribution.sh</path>
          <type>arch_sh</type>
        </item>
        <item>
          <path>/usr/local/share/vzlinmigrate</path>
          <type>dir</type>
        </item>
        <dst_path>/var/vzagent.tmp</dst_path>
      </distribution>
      <distribution>
        <architecture>x86_64</architecture>
        <item>
          <path>/opt/vzagent/distribution/distribution-x86_64.sh</path>
          <type>arch_sh</type>
        </item>
        <item>
          <path>/usr/local/share/vzlinmigrate</path>
          <type>dir</type>
        </item>
        <dst_path>/var/vzagent.tmp</dst_path>
      </distribution>
      <distribution>
        <architecture>ia64</architecture>
        <item>
          <path>/opt/vzagent/distribution/distribution-ia64.sh</path>
          <type>arch_sh</type>
        </item>
        <item>
          <path>/usr/local/share/vzlinmigrate</path>
          <type>dir</type>
        </item>
        <dst_path>/var/vzagent.tmp</dst_path>
      </distribution>
      <timeouts>
        <distribute>1800</distribute>
        <connectivity_check>300</connectivity_check>
      </timeouts>
      <map>
        <ns1:node>
          <ns1:id>638a2a56-e689-c340-877d-bd0470f2448c</ns1:id>
```

```
        <ns1:host>local</ns1:host>
        <ns1:conn_info xsi:type="ns2:connection_infoType">
          <ns2:protocol>SSL</ns2:protocol>
          <ns2:port>4434</ns2:port>
          <ns2:login xsi:type="ns2:auth_nameType"/>
          <ns2:address>local</ns2:address>
        </ns1:conn_info>
      </ns1:node>
    </map>
    <log>off</log>
    <ns1:id>638a2a56-e689-c340-877d-bd0470f2448c</ns1:id>
  </configuration>
</system>
<gend>
  <configuration>
    <operation_timing/>
    <default_pool>
      <timeout_limit>300</timeout_limit>
      <heavy_timeout_limit>360000</heavy_timeout_limit>
      <urgent_timeout_limit>60</urgent_timeout_limit>
      <queue>50</queue>
      <pool>10</pool>
      <heavy_pool>4</heavy_pool>
      <urgent_pool>20</urgent_pool>
      <comeback_ratio>4</comeback_ratio>
      <default_timeout>300</default_timeout>
      <kill_timeout>20</kill_timeout>
    </default_pool>
    <default_single>
      <queue>50</queue>
    </default_single>
    <server_group>
      <queue>500</queue>
    </server_group>
    <client>
      <queue>30</queue>
    </client>
    <default_remote>
      <default_timeout>360000</default_timeout>
      <connect_timeout>20</connect_timeout>
      <startup_timeout>10</startup_timeout>
      <child_connect_timeout>20</child_connect_timeout>
      <child_inactivity_timeout>300</child_inactivity_timeout>
    </default_remote>
  </configuration>
</gend>
<res_log>
  <configuration>
    <start_time>00:00</start_time>
    <compress>
      <interval>86400</interval>
      <offset>604800</offset>
      <duration>604800</duration>
      <period>14400</period>
      <start_delay>60</start_delay>
    </compress>
    <compress>
      <interval>604800</interval>
      <offset>2592000</offset>
      <duration>2592000</duration>
      <period>86400</period>
      <start_delay>3610</start_delay>
    </compress>
    <compress>
      <interval>2592000</interval>
      <offset>15552000</offset>
      <duration>15552000</duration>
```

```
        <period>2592000</period>
        <start_delay>7220</start_delay>
      </compress>
    </configuration>
  </res_log>
  <servicem>
    <configuration>
      <timeouts>
        <operate>120</operate>
      </timeouts>
    </configuration>
  </servicem>
  <relocator>
    <configuration>
      <default_migration_type>0</default_migration_type>
    </configuration>
  </relocator>
  <envcoll>
    <configuration>
      <timeouts>
        <refresh>120</refresh>
      </timeouts>
    </configuration>
  </envcoll>
  <pager>
    <configuration/>
  </pager>
  <vzlpagercache>
    <configuration>
      <timeouts>
        <refresh>120</refresh>
      </timeouts>
    </configuration>
  </vzlpagercache>
  <packagem>
    <configuration/>
  </packagem>
  <securitym>
    <configuration>
      <enable>true</enable>
      <az_store>
        <port>389</port>
      </az_store>
    </configuration>
  </securitym>
  <ns3:authm>
    <ns3:configuration>
      <ns3:realms>
        <ns3:realm xsi:type="ns3:dir_realmType">
          <ns2:id>c9c47ec6-bbd0-1445-bc50-3cb72179cad7</ns2:id>
          <ns2:type>1</ns2:type>
          <ns2:name>Virtuozzo Internal</ns2:name>
          <ns3:address>vzsveaddress</ns3:address>
          <ns3:port>389</ns3:port>
          <ns3:base_dn>dc=vzl</ns3:base_dn>
          <ns3:default_dn>cn=users,dc=vzl</ns3:default_dn>
          <ns3:login>
            <ns2:name>Y249dnphZ2VudCxkYz1WWkw=</ns2:name>
          </ns3:login>
        </ns3:realm>
      </ns3:realms>
      <ns3:default_realm>c9c47ec6-bbd0-1445-bc50-
3cb72179cad7</ns3:default_realm>
      <ns3:builtin_realm>c9c47ec6-bbd0-1445-bc50-
3cb72179cad7</ns3:builtin_realm>
    </ns3:configuration>
  </ns3:authm>
```

```
<sessionm>
  <configuration>
    <session_expiration>10800</session_expiration>
    <persistent_session_expiration>180</persistent_session_expiration>
  </configuration>
</sessionm>
<http_configurator>
  <configuration>
    <paths>
      templateredhat</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <paths>
      templatesuse</template>
      <config>/etc/apache2/httpd.conf</config>
      <bin>/usr/sbin/httpd2</bin>
      <service>/etc/init.d/apache2</service>
    </paths>
    <paths>
      templatedebian</template>
      <config>/etc/apache/httpd.conf</config>
      <bin>/usr/sbin/apache</bin>
      <service>/etc/init.d/apache</service>
    </paths>
    <paths>
      templatefedora core</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <paths>
      templatecentos-3</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <paths>
      templatecentos-4</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <timeouts>
      <analize>1800</analize>
    </timeouts>
    <config_limit>1000000</config_limit>
  </configuration>
</http_configurator>
<env_sample_monitor>
  <configuration>
    <sync_period>1800</sync_period>
  </configuration>
</env_sample_monitor>
<env_samplem>
  <configuration/>
</env_samplem>
<processm>
  <configuration/>
</processm>
<userm>
  <configuration/>
</userm>
<computerm>
  <configuration>
    <timeouts>
```

```
      <log>120</log>
      <migrate_key>60</migrate_key>
      <migrate>3600</migrate>
    </timeouts>
  </configuration>
</computerm>
<scheduler>
  <configuration/>
</scheduler>
<devm>
  <configuration/>
</devm>
<env_event_mon>
  <configuration>
    <period>10</period>
  </configuration>
</env_event_mon>
<envm>
  <configuration>
    <timeouts>
      <create>3600</create>
      <operate>300</operate>
      <init>1200</init>
      <clone>3600</clone>
      <move>3600</move>
    </timeouts>
  </configuration>
</envm>
<server_group>
  <configuration>
    <realm>c9c47ec6-bbd0-1445-bc50-3cb72179cad7</realm>
  </configuration>
</server_group>
<resourcem>
  <configuration/>
</resourcem>
<vzanetworkm>
  <configuration/>
</vzanetworkm>
<firewallm>
  <configuration/>
</firewallm>
<perf_mon>
  <configuration/>
</perf_mon>
<vzasysd>
  <configuration>
    <cpu_check_period>300</cpu_check_period>
  </configuration>
</vzasysd>
<op_log>
  <configuration>
    <events>604800</events>
    <operation_log>604800</operation_log>
  </configuration>
</op_log>
<resource_alert_monitor>
  <configuration>
    <period>10</period>
  </configuration>
</resource_alert_monitor>
<alertm>
  <configuration>
    <mute_alert_period>-1</mute_alert_period>
    <support_email>support@localhost</support_email>
  </configuration>
</alertm>
```

```
<backupm>
  <configuration>
    <timeouts>
      <backup>3600</backup>
    </timeouts>
  </configuration>
</backupm>
<backup_storagem>
  <configuration>
    <timeouts>
      <notify>30</notify>
    </timeouts>
  </configuration>
</backup_storagem>
<filer>
  <configuration/>
</filer>
<mailer>
  <configuration/>
</mailer>
<vza_env_sample_monitor>
  <configuration>
    <sync_period>30</sync_period>
  </configuration>
</vza_env_sample_monitor>
<vzatbs>
  <configuration>
    <services>
      <service>httpd</service>
      <service>sendmail</service>
      <service>sshd</service>
    </services>
    <timeouts>
      <template_consistency>600</template_consistency>
      <vzfs_check>1200</vzfs_check>
      <diskquota_fix>1200</diskquota_fix>
      <template_verify>3600</template_verify>
      <template_recover>1200</template_recover>
    </timeouts>
  </configuration>
</vzatbs>
<vza_conf>
  <configuration>
    <check_period>5</check_period>
    <sync_period>60</sync_period>
    <dhcp_refresh_min>300</dhcp_refresh_min>
    <dhcp_refresh_max>3600</dhcp_refresh_max>
    <skip_update>0</skip_update>
  </configuration>
</vza_conf>
<vza_perf>
  <configuration>
    <periods>
      <counters_vz_quota>20</counters_vz_quota>
      <counters_disk>20</counters_disk>
    </periods>
  </configuration>
</vza_perf>
<vzaenvm>
  <configuration>
    <start_veid>100</start_veid>
    <end_veid>9999999</end_veid>
    <sve_visible>0</sve_visible>
    <timeouts>
      <create>3600</create>
      <operate>300</operate>
      <init>1200</init>
```

```
        <clone>3600</clone>
        <move>3600</move>
      </timeouts>
    </configuration>
  </vzaenvm>
  <vzapackagemonitor>
    <configuration>
      <timeouts>
        <refresh>121</refresh>
      </timeouts>
    </configuration>
  </vzapackagemonitor>
  <vzapackagem>
    <configuration/>
  </vzapackagem>
</data>
</ns1:packet>
```

connect

Summary:

Establishes an exclusive connection with a remote Agent.

Request specification:

Name	Min/Max	Type	Description
`connect`			
`{`			
`conn_info`	`1..1`	`connection_infoType (p. 30)`	The remote server connection information.
`}`			

Returns:

Name	Min/Max	Type	Description
`id`	`0..1`	`string`	The ID of the connection that was established to the remote server. When sending requests to the remote server, use this ID as the value of the `target` parameter in the message header. The target Agent operator must be supplied using the `dst/target` parameter (see code examples below).

Description:

The `connect` call allows to establish an exclusive connection between the Agent that you are currently connected to and the Agent installed on a remote server. The connection can then be used to route the messages from the current server to the remote server. Once the connection is established, it can be used infinitely with one exception. The connection times out and closes automatically in 10 minutes of inactivity. To keep the connection open without actually using it, it is enough to ping the remote server over it periodically using the `ping` call (p. 555). When the connection is no longer needed, use `close` (p. 517) to terminate it.

Example:

Establishing an exclusive connection with the specified remote Agent.

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <connect>
        <conn_info>
          <protocol>SSL</protocol>
          <address>192.168.0.84</address>
          <login>
            <name>cm9vdA==</name>
            <realm>00000000-0000-0000-0000-000000000000</realm>
          </login>
          <password>bXlwYXNz</password>
          <port>4434</port>
        </conn_info>
      </connect>
    </system>
  </data>
</packet>
```

Output

The output contains the connection ID.

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="21c466e9712t41bbra18" priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient5-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:id>192.168.0.841</ns2:id>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

The returned connection ID can be used to send Agent requests to the remote server as shown in the following example.

Input

Retrieving a list of Virtuozzo Containers from the remote server. In order to route the messages to the remote server, the `target` element contains the connection ID that we obtained in the previous step. The `dst/target` element contains the name of the target Agent operator (the `vzaenvm` operator in our case). The `data` element is composed in a usual manner.

```
<packet version="4.0.0">
  <target>192.168.0.841</target>
  <dst>
    <target>vzaenvm</target>
  </dst>
  <data>
    <vzaenvm>
      <get_list/>
    </vzaenvm>
  </data>
</packet>
```

The following example pings the remote server over the exclusive connection that we established earlier. Doing so will reset the timeout timer and will keep the connection open.

Input

```
<packet version="4.0.0">
  <target>192.168.0.841</target>
  <data>
    <system>
      <ping/>
    </system>
  </data>
</packet>
```

count_registered

Summary:

Provides information about the clients that are currently registered with the Agent. Used with the `register_client` call (p. 557) to implement licensing functionality in the client software.

Request specification:

Name	Min/Max	Type	Description
`count_registered`			
`{`			
`id`	`0..1`	`string`	The ID of the client to retrieve the connection info for. If the element is omitted, the information for all registered clients will be retrieved. The ID is assigned to the client connection at the time it is registered with the Agent using the `register_client` call (p. 557).
`}`			

Returns:

Name	Min/Max	Type	Description
registered	0..[]		Client connection information.
{			
id	1..1	string	
count	1..1	int	
}			

Description:

The `count_registered` call (p. 527) and the `register_client` call (p. 557) allow keeping track of the logged in clients and limiting the number of concurrent connections from the same client application to a given Agent. The following describes a typical usage scenario.

As soon as a client establishes a connection with the Agent, use the `count_registered` call (p. 527) to get the number of currently registered clients with the same ID. Depending on the result, one of the following should happen:

- If the number is less then the maximum allowed number of concurrent connections (the maximum number is determined by you) the login is granted. The new connection is then registered with the Agent using the `register_client` call (p. 557).
- If the number is equal to or greater than the maximum allowed number of connections, the login is denied and the connection is terminated.

It is not necessary to unregister the connection when the client logs off, as Agent does that automatically.

Please note that this call is used to count the permanent connections only. To count the user sessions, use the `sessionm/count_registered` call (p. 481).

Example:

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <count_registered/>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="51c467017bct323bra18" priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient62-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:registered>
      <ns2:id>license_id_333</ns2:id>
      <ns2:count>1</ns2:count>
    </ns2:registered>
    <ns2:registered>
      <ns2:id>license_id_334</ns2:id>
      <ns2:count>1</ns2:count>
    </ns2:registered>
    <ns2:registered>
      <ns2:id>license_id_335</ns2:id>
      <ns2:count>2</ns2:count>
```

```
    </ns2:registered>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

distribute

Summary:

Distributes Agent core to a server that doesn't have Virtuozzo Containers software installed on it.

Request specification:

Name	Min/Max	Type	Description
distribute			
{			
dst	1..1	connection_infoType (p. 30)	The target node connection information.
}			

Returns:

Name	Min/Max	Type	Description
eid	0..1	eid_type (p. 25)	The Server ID that was assigned to the target node.

Description:

The `distribute` call uploads Agent core to a server, starts the uploaded Agent, and tries to establish a connection with it. If all of the above steps complete successfully, the call returns a Server ID identifying the target server. The distribution of Agent core is a necessary step during some of the Agent operations, such as the physical-to-virtual migration (p. 383). The returned Server ID can also be used to route Agent messages to the server through another Node in a Virtuozzo group.

Example:

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <distribute>
        <dst>
          <protocol>TCP</protocol>
          <address>192.168.0.123</address>
          <login>
            <name>cm9vdA==</name>
          </login>
          <password>bXlwYXNz</password>
          <port>4433</port>
        </dst>
      </distribute>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="fc465a9994t5f90rc38"
time="2007-05-24T13:26:54+0000" priority="4000" version="4.0.0">
<ns1:origin>system</ns1:origin>
<ns1:target>vzclient21-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:system>
    <ns2:eid>a27a53ef-ef7c-41df-8704-7bfd5023c30c</ns2:eid>
  </ns2:system>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

generate_pass

Summary:

Generate a one-time login info that can be used to establish a connection with the specified server.

Request specification:

Name	Min/Max	Type	Description
generate_pass			
{			
eid	0..1	eid_type (p. 25)	
}			

Returns:

Name	Min/Max	Type	Description
pass	1..1	auth_nameType (p. 29)	

Description:

The login info generated by the `generate_pass` call can be used to establish a connection with the specified server only once. Use this call when you want to establish a connection with a server in order to perform some task but don't want to send your permanent user ID and password over the network. You would normally use a one-time login info to establish a temporary connection with a remote Hardware Node to perform an operation such as server migration. On operation completion, the temporary connection is automatically terminated. At the same time the login info that was used to connect to the remote machine becomes invalid and cannot be used again.

Example:

The following example generates the temporary login info for the remote Hardware Node specified in the `dst` element in the message header.

Input

```
<packet version="4.0.0">
  <dst>
    Host08226eb6-113a-1045-8716-e738d669fd4e</host>
  </dst>
  <data>
    <system>
      <generate_pass/>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/sessionm"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="3ec4655918et4509rd64" priority="0" version="4.0.0">
<ns1:origin>system</ns1:origin>
<ns1:target>vzclient7-1df4b04e-0d55-f246-b718-89bbc62fd371</ns1:target>
<ns1:data>
  <ns1:system>
    <ns2:pass xsi:type="ns3:auth_nameType">
      <ns3:domain>bWFzdGVy</ns3:domain>
      <ns3:name>MzVhYWRjYjYtZGFmNS1lOTQ1LTk0MjQtNzdhMDczN2ExZGNh</ns3:name>
      <ns3:realm>00000000-0000-0000-0000-000000000002</ns3:realm>
    </ns2:pass>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

get_configuration

Summary:

Retrieves the Agent configuration information.

Request specification:

Name	Min/Max	Type	Description
`get_configuration`	`1..1`	`none`	

Returns:

The complete Agent configuration information.

Description:

The Agent configuration data structure consists of individual segments each describing a specific Agent operator or director settings. Some of the settings deal with the internal workings of Agent and will probably be of no interest to an average application developer (these settings should never be modified). Other settings may directly affect the way your client programs operate. Specifically, you might want to review the preset timeout limits and adjust them to suit your particular situation if needed. To modify the Agent configuration, use the `configuration` call (p. 518).

Example:

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <get_configuration/>
    </system>
  </data>
</packet>
```

Output

```
<packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="19c46b87e07t2cd6rf28" priority="0" version="4.0.0">
<origin>gend</origin>
<target>vzclient3-5850cc75-cbde-784c-be21-026fcd46c9d7</target>
<dst>
  <director>gend</director>
</dst>
<data>
  <system>
    <configuration>
      <distribution>
        <item>
          <path>/opt/vzagent/distribution/distribution.sh</path>
          <type>arch_sh</type>
        </item>
        <item>
          <path>/usr/local/share/vzlinmigrate</path>
          <type>dir</type>
        </item>
        <dst_path>/var/vzagent.tmp</dst_path>
      </distribution>
      <distribution>
        <architecture>x86_64</architecture>
        <item>
          <path>/opt/vzagent/distribution/distribution-x86_64.sh</path>
          <type>arch_sh</type>
        </item>
        <item>
          <path>/usr/local/share/vzlinmigrate</path>
          <type>dir</type>
        </item>
        <dst_path>/var/vzagent.tmp</dst_path>
      </distribution>
```

```
    <distribution>
      <architecture>ia64</architecture>
      <item>
        <path>/opt/vzagent/distribution/distribution-ia64.sh</path>
        <type>arch_sh</type>
      </item>
      <item>
        <path>/usr/local/share/vzlinmigrate</path>
        <type>dir</type>
      </item>
      <dst_path>/var/vzagent.tmp</dst_path>
    </distribution>
    <timeouts>
      <distribute>1800</distribute>
      <connectivity_check>300</connectivity_check>
    </timeouts>
    <map>
      <node>
        <id>5850cc75-cbde-784c-be21-026fcd46c9d7</id>
        Hostlocal</host>
        <conn_info xsi:type="ns1:connection_infoType">
          <protocol>SSL</protocol>
          <port>4434</port>
          <login xsi:type="ns1:auth_nameType">
            <realm>00000000-0000-0000-0000-000000000000</realm>
          </login>
          <address>local</address>
        </conn_info>
      </node>
    </map>
    <log>off</log>
    <id>5850cc75-cbde-784c-be21-026fcd46c9d7</id>
  </configuration>
</system>
<gend>
  <configuration>
    <operation_timing/>
    <default_pool>
      <timeout_limit>300</timeout_limit>
      <heavy_timeout_limit>360000</heavy_timeout_limit>
      <urgent_timeout_limit>60</urgent_timeout_limit>
      <queue>50</queue>
      <pool>10</pool>
      <heavy_pool>4</heavy_pool>
      <urgent_pool>20</urgent_pool>
      <comeback_ratio>4</comeback_ratio>
      <default_timeout>300</default_timeout>
      <kill_timeout>20</kill_timeout>
    </default_pool>
    <default_single>
      <queue>50</queue>
    </default_single>
    <server_group>
      <queue>500</queue>
    </server_group>
    <client>
      <queue>30</queue>
    </client>
    <default_remote>
      <default_timeout>360000</default_timeout>
      <connect_timeout>20</connect_timeout>
      <startup_timeout>10</startup_timeout>
      <child_connect_timeout>20</child_connect_timeout>
      <child_inactivity_timeout>300</child_inactivity_timeout>
    </default_remote>
  </configuration>
</gend>
```

```
<vzlin_backup_serializer>
  <configuration>
    <backend>0</backend>
  </configuration>
</vzlin_backup_serializer>
<vzaenvm>
  <configuration>
    <start_veid>100</start_veid>
    <end_veid>9999999</end_veid>
    <sve_visible>0</sve_visible>
    <timeouts>
      <create>3600</create>
      <operate>300</operate>
      <init>1200</init>
      <clone>3600</clone>
      <move>3600</move>
    </timeouts>
  </configuration>
</vzaenvm>
<vzatbs>
  <configuration>
    <services>
      <service>httpd</service>
      <service>sendmail</service>
      <service>sshd</service>
    </services>
    <timeouts>
      <template_consistency>600</template_consistency>
      <vzfs_check>1200</vzfs_check>
      <diskquota_fix>1200</diskquota_fix>
      <template_verify>3600</template_verify>
      <template_recover>1200</template_recover>
    </timeouts>
  </configuration>
</vzatbs>
<vza_conf>
  <configuration>
    <check_period>5</check_period>
    <sync_period>60</sync_period>
    <dhcp_refresh_min>60</dhcp_refresh_min>
    <dhcp_refresh_max>3600</dhcp_refresh_max>
    <skip_update>0</skip_update>
  </configuration>
</vza_conf>
<vza_env_sample_monitor>
  <configuration>
    <sync_period>30</sync_period>
  </configuration>
</vza_env_sample_monitor>
<vza_perf>
  <configuration>
    <periods>
      <counters_vz_common>20</counters_vz_common>
      <counters_vz_net>20</counters_vz_net>
      <counters_vz_process>20</counters_vz_process>
      <counters_vz_quota>20</counters_vz_quota>
      <counters_disk>20</counters_disk>
    </periods>
  </configuration>
</vza_perf>
<vzanetworkm>
  <configuration/>
</vzanetworkm>
<vzapackagem>
  <configuration/>
</vzapackagem>
<vzapackagemonitor>
```

```
    <configuration>
      <timeouts>
        <refresh>120</refresh>
      </timeouts>
    </configuration>
  </vzapackagemonitor>
  <vzasysd>
    <configuration>
      <cpu_check_period>300</cpu_check_period>
    </configuration>
  </vzasysd>
  <vzaup2date>
    <configuration>
      <no_signatures>0</no_signatures>
    </configuration>
  </vzaup2date>
  <alertm>
    <configuration>
      <mute_alert_period>-1</mute_alert_period>
      <support_email>support@localhost</support_email>
    </configuration>
  </alertm>
  <resource_alert_monitor>
    <configuration>
      <period>60</period>
    </configuration>
  </resource_alert_monitor>
  <sessionm>
    <configuration>
      <session_expiration>10800</session_expiration>
      <persistent_session_expiration>180</persistent_session_expiration>
    </configuration>
  </sessionm>
  <authm>
    <configuration>
      <realms>
        <realm xsi:type="ns2:dir_realmType">
          <id>d4e56e5a-631a-484c-8173-6b50cc43d1d1</id>
          <type>1</type>
          <name>Virtuozzo Internal</name>
          <address>vzsveaddress</address>
          <port>389</port>
          <base_dn>ou=5850cc75-cbde-784c-be21-026fcd46c9d7,dc=vzl</base_dn>
          <default_dn>cn=users,ou=5850cc75-cbde-784c-be21-
026fcd46c9d7,dc=vzl</default_dn>
          <login>
            <name>Y249dnphZ2VudCxkYz1WWkw=</name>
            <realm>d4e56e5a-631a-484c-8173-6b50cc43d1d1</realm>
          </login>
        </realm>
      </realms>
      <default_realm>d4e56e5a-631a-484c-8173-6b50cc43d1d1</default_realm>
      <builtin_realm>d4e56e5a-631a-484c-8173-6b50cc43d1d1</builtin_realm>
    </configuration>
  </authm>
  <securitym>
    <configuration>
      <az_store>
        <port>389</port>
      </az_store>
      <timeout>600</timeout>
    </configuration>
  </securitym>
  <backup_deserializer>
    <configuration>
      <port>4435</port>
      <default_pool>
```

```
      <heavy_timeout_limit>3600</heavy_timeout_limit>
    </default_pool>
  </configuration>
</backup_deserializer>
<restore_serializer>
  <configuration>
    <port>4435</port>
    <default_pool>
      <heavy_timeout_limit>3600</heavy_timeout_limit>
    </default_pool>
  </configuration>
</restore_serializer>
<backup_storagem>
  <configuration>
    <timeouts>
      <notify>30</notify>
    </timeouts>
  </configuration>
</backup_storagem>
<backupm>
  <configuration>
    <timeouts>
      <backup>3600</backup>
    </timeouts>
  </configuration>
</backupm>
<envm>
  <configuration>
    <timeouts>
      <create>3600</create>
      <operate>300</operate>
      <init>1200</init>
      <clone>3600</clone>
      <move>3600</move>
    </timeouts>
  </configuration>
</envm>
<server_group>
  <configuration>
    <realm>d4e56e5a-631a-484c-8173-6b50cc43d1d1</realm>
  </configuration>
</server_group>
<env_event_mon>
  <configuration>
    <period>10</period>
  </configuration>
</env_event_mon>
<computerm>
  <configuration>
    <timeouts>
      <log>120</log>
      <migrate_key>60</migrate_key>
      <migrate>3600</migrate>
    </timeouts>
  </configuration>
</computerm>
<devm>
  <configuration/>
</devm>
<env_sample_monitor>
  <configuration>
    <sync_period>1800</sync_period>
  </configuration>
</env_sample_monitor>
<env_samplem>
  <configuration/>
</env_samplem>
```

```
<filer>
  <configuration/>
</filer>
<firewallm>
  <configuration/>
</firewallm>
<http_configurator>
  <configuration>
    <paths>
      templateredhat</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <paths>
      templatesuse</template>
      <config>/etc/apache2/httpd.conf</config>
      <bin>/usr/sbin/httpd2</bin>
      <service>/etc/init.d/apache2</service>
    </paths>
    <paths>
      templatedebian</template>
      <config>/etc/apache/httpd.conf</config>
      <bin>/usr/sbin/apache</bin>
      <service>/etc/init.d/apache</service>
    </paths>
    <paths>
      templatefedora core</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <paths>
      templatecentos-3</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <paths>
      templatecentos-4</template>
      <config>/etc/httpd/conf/httpd.conf</config>
      <bin>/usr/sbin/httpd</bin>
      <service>/etc/init.d/httpd</service>
    </paths>
    <timeouts>
      <analize>1800</analize>
    </timeouts>
    <config_limit>1000000</config_limit>
  </configuration>
</http_configurator>
<op_log>
  <configuration>
    <events>604800</events>
    <operation_log>604800</operation_log>
  </configuration>
</op_log>
<mailer>
  <configuration/>
</mailer>
<migrator>
  <configuration>
    <default_migration_type>0</default_migration_type>
  </configuration>
</migrator>
<packagem>
  <configuration/>
</packagem>
```

```
  <pager>
    <configuration/>
  </pager>
  <envcoll>
    <configuration>
      <timeouts>
        <refresh>120</refresh>
      </timeouts>
    </configuration>
  </envcoll>
  <vzlpagercache>
    <configuration>
      <timeouts>
        <refresh>120</refresh>
      </timeouts>
    </configuration>
  </vzlpagercache>
  <perf_mon>
    <configuration/>
  </perf_mon>
  <processm>
    <configuration/>
  </processm>
  <res_log>
    <configuration>
      <start_time>00:00</start_time>
      <compress>
        <interval>86400</interval>
        <offset>604800</offset>
        <duration>604800</duration>
        <period>14400</period>
        <start_delay>60</start_delay>
      </compress>
      <compress>
        <interval>604800</interval>
        <offset>2592000</offset>
        <duration>2592000</duration>
        <period>86400</period>
        <start_delay>3610</start_delay>
      </compress>
      <compress>
        <interval>2592000</interval>
        <offset>15552000</offset>
        <duration>15552000</duration>
        <period>2592000</period>
        <start_delay>7220</start_delay>
      </compress>
    </configuration>
  </res_log>
  <resourcem>
    <configuration/>
  </resourcem>
  <scheduler>
    <configuration/>
  </scheduler>
  <servicem>
    <configuration>
      <timeouts>
        <operate>120</operate>
      </timeouts>
    </configuration>
  </servicem>
  <userm>
    <configuration/>
  </userm>
</data>
</packet>
```

get_eid

Summary:

Returns the Server ID of the Hardware Node you are currently connected to.

Request specification:

Name	Min/Max	Type	Description
get_eid	1..1	none	

Returns:

Name	Min/Max	Type	Description
eid	0..1	eid_type (p. 25)	The Server ID assigned to the Hardware Node by Agent.

Example:

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <get_eid/>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="bc467112fbt3d6cr81c"
priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient3-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:eid>638a2a56-e689-c340-877d-bd0470f2448c</ns2:eid>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

get_plugins

Summary:

Retrieves a list of installed Agent plug-ins.

Request specification:

Name	Min/Max	Type	Description
get_plugins	1..1		

Returns:

Name	Min/Max	Type	Description
`plugin`	`0..[]`		
`{`			
`  name`	`1..1`	`string`	The name of the plug-in.
`}`			

Description:

If you are experiencing problems using one of the Agent functions, you can use the `get_plugins` call to verify that the appropriate plug-in module is installed on the server. If you don't see the module in the list, you may try reinstalling Agent on the server.

Example:

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <get_plugins/>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="cc46711f0et2cd6r81c"
priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient3-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:plugin>
      <ns2:name>VZABackupManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZABasicFunctionality</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZARelocator</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZANetworkManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZAOpCompat</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZAPackageManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZAServCompat</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZASysD</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZAUp2date</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLAlertManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLAuthEngine</ns2:name>
```

```
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLBackupManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLCluster</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLComputerManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLDeviceManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLEnvSampleManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLFileManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLFirewallManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLHTTPConfigurator</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLLicenseManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLLogDB</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLRelocator</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLPackageManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLPager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLPerformanceCollector</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLPerformanceMonitor</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLProcessManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLResLog</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLResourceManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLScheduler</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLServiceManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZLUserManager</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>VZVBasicFunctionality</ns2:name>
    </ns2:plugin>
```

```
    <ns2:plugin>
      <ns2:name>mailer</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>vzaproxy</ns2:name>
    </ns2:plugin>
    <ns2:plugin>
      <ns2:name>vzaproxyinsve</ns2:name>
    </ns2:plugin>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

get_realm

Summary:

Retrieves the list of the available Realms from the current Agent configuration.

Request specification:

Name	Min/Max	Type	Description
get_realm	1..1	none	

Returns:

Name	Min/Max	Type	Description
realms	1..1		
{			
realm	1..[]	realmType (p. 54)	The list of the available realms. The appropriate data type will be used for a particular realm type (LDAP, System, etc.). See the subtypes of the realmType type for the available realm types.
}			

Description:

The `get_realm` request can be executed without being logged in to Agent. It's purpose is to get the IDs of the available realms to use in the `system/login` call (p. 553).

Example:

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <get_realm/>
    </system>
  </data>
</packet>
```

Output

```
<packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/authm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="8c46b82a5at18berf28"
priority="0" version="4.0.0">
<origin>system</origin>
<target>vzclient4-2cbbe469-8f57-7f46-97fb-fd987231d957</target>
<data>
  <system>
    <realms>
      <realm xsi:type="ns2:dir_realmType">
        <login>
          <name>Y249dnphZ2VudCxkYz1WWkw=</name>
          <realm>3e761571-6607-1344-a064-a42679da8ed9</realm>
        </login>
        <builtin/>
        <name>Virtuozzo Internal</name>
        <type>1</type>
        <id>3e761571-6607-1344-a064-a42679da8ed9</id>
        <address>vzsveaddress</address>
        <port>389</port>
        <base_dn>dc=vzl</base_dn>
        <default_dn>cn=users,dc=vzl</default_dn>
      </realm>
      <realm xsi:type="ns3:realmType">
        <builtin/>
        <name>System</name>
        <type>0</type>
        <id>00000000-0000-0000-0000-000000000000</id>
      </realm>
      <realm xsi:type="ns3:realmType">
        <builtin/>
        <name>Virtuozzo Container</name>
        <type>1000</type>
        <id>00000000-0000-0000-0100-000000000000</id>
      </realm>
    </realms>
  </system>
</data>
</packet>
```

get_state

Summary:

Gets the current states of Agent operators.

Request specification:

Name	Min/Max	Type	Description
get_state	1..1	none	

Returns:

Name	Min/Max	Type	Description
state	0..[]		A list of operators.
{			
operator	1..1	string	Operator name.
queue_size	1..1	int	The size of the queue.
queue_limit	1..1	int	The queue limit.
pool_total	0..1	int	The pool size.
pool_free	0..1	int	The number of operators that are currently available in the pool.
pool_busy	0..1	int	The number of operators that are currently busy.
pool_heavy	0..1	int	The number of operators that are busy processing "heavy" messages.
normal_dyn_limit	0..1	float	Normal messages dynamic limit.
heavy_dyn_limit	0..1	float	Heavy messages dynamic limit.
urgent_dyn_limit	0..1	float	Urgent messages dynamic limit.
normal_limit	0..1	int	Normal messages total limit.
heavy_limit	0..1	int	Heavy messages total limit.
urgent_limit	0..1	int	Urgent messages total limit.
timeout	0..1	int	The default timeout value for the operator.
kill_timeout	0..1	int	The timeout value after which the process associated with the operator will be killed if the default timeout (the timeout parameter above) has already happened.

timings	0..1		This structure contains the timing values for the individual operations associated with the given operator. The values can be used for performance evaluation while optimizing the client code.
{			
timing	0..[]		A list of operations and their timing values.
{			
operation	1..1	string	The name of the operation.
count	1..1	int	The total number of times the operation has been executed since the Agent was started on the current machine.
avg	1..1	int	The average time per instance the operation has taken to execute.
min	1..1	int	The minimum execution time.
max	1..1	int	The maximum execution time.
time	1..1	long	The execution time of all the invocation instances combined.
}			
}			
}			

Example:

Input

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="dc467128aet72aer81c"
priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient3-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:state>
      <ns2:operator>alertm</ns2:operator>
      <ns2:queue_size>0</ns2:queue_size>
      <ns2:queue_limit>50</ns2:queue_limit>
      <ns2:timings/>
    </ns2:state>
    <ns2:state>
      <ns2:operator>authm</ns2:operator>
      <ns2:queue_size>0</ns2:queue_size>
      <ns2:queue_limit>50</ns2:queue_limit>
      <ns2:timings>
        <ns2:timing>
          <ns2:operation>authenticate</ns2:operation>
          <ns2:count>3</ns2:count>
          <ns2:avg>9</ns2:avg>
          <ns2:min>5</ns2:min>
          <ns2:max>15</ns2:max>
          <ns2:time>29</ns2:time>
        </ns2:timing>
      </ns2:timings>
    </ns2:state>
    <ns2:state>
      <ns2:operator>backup_deserializer</ns2:operator>
      <ns2:queue_size>0</ns2:queue_size>
      <ns2:queue_limit>50</ns2:queue_limit>
      <ns2:pool_total>0</ns2:pool_total>
      <ns2:pool_free>0</ns2:pool_free>
      <ns2:pool_busy>0</ns2:pool_busy>
      <ns2:pool_heavy>0</ns2:pool_heavy>
      <ns2:normal_dyn_limit>10.000000</ns2:normal_dyn_limit>
      <ns2:heavy_dyn_limit>4.000000</ns2:heavy_dyn_limit>
      <ns2:urgent_dyn_limit>20.000000</ns2:urgent_dyn_limit>
      <ns2:normal_limit>10</ns2:normal_limit>
      <ns2:heavy_limit>4</ns2:heavy_limit>
      <ns2:timeout>300</ns2:timeout>
      <ns2:kill_timeout>20</ns2:kill_timeout>
      <ns2:timings/>
    </ns2:state>
    <ns2:state>
      <ns2:operator>backup_storagem</ns2:operator>
      <ns2:queue_size>0</ns2:queue_size>
      <ns2:queue_limit>50</ns2:queue_limit>
      <ns2:pool_total>0</ns2:pool_total>
      <ns2:pool_free>0</ns2:pool_free>
      <ns2:pool_busy>0</ns2:pool_busy>
      <ns2:pool_heavy>0</ns2:pool_heavy>
      <ns2:normal_dyn_limit>10.000000</ns2:normal_dyn_limit>
      <ns2:heavy_dyn_limit>4.000000</ns2:heavy_dyn_limit>
      <ns2:urgent_dyn_limit>20.000000</ns2:urgent_dyn_limit>
      <ns2:normal_limit>10</ns2:normal_limit>
```

```
      <ns2:heavy_limit>4</ns2:heavy_limit>
      <ns2:timeout>300</ns2:timeout>
      <ns2:kill_timeout>20</ns2:kill_timeout>
      <ns2:timings/>
    </ns2:state>

    <!-- The rest of the output is omitted for brevity -->
  </ns1:system>
</ns1:data>
</ns1:packet>
```

get_version

Summary:

Returns the internal Agent version number. Please note that this number is not the same as the protocol number (which is defined in the XML schema and is 4.0.0 at the time of this writing). Different versions of Agent software may use the same protocol version number so the two numbers are not directly linked.

Request specification:

Name	Min/Max	Type	Description
get_version	1..1	none	

Returns:

Name	Min/Max	Type	Description
version	0..1	string	The internal Agent version number.

Example:

Input

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <get_version/>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="ec46712dddt6952r81c"
priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient3-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:version>vzagent-4.0.0</ns2:version>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

get_vocabulary

Summary:

Retrieves the Agent vocabulary data. The call parameters can be used to perform selective retrieval of the vocabulary entries.

Request specification:

Name	Min/Max	Type	Description
`get_vocabulary`			
`{`			
`name`	`0..[]`	`string`	The name of the plug-in. There's a separate section in the vocabulary for every Agent plug-in. By specifying the plug-in name here, you will limit the search to that section only.
`parameter`	`0..[]`	`string`	The name of the parameter in the vocabulary. if you know the name of the parameter that you are looking for, you may specify it here. The information will be retrieved for that parameter only. If the `category` parameter (below) is also specified, the call will search for the parameter in that category only.
`category`	`0..[]`	`string`	The name of the category in the vocabulary. If specified, the search will be limited to that category only.
`}`			

Returns:

Name	Min/Max	Type	Description
`parameter`	`0..[]`	`voc_parameterType (p. 514)`	The requested Agent vocabulary data. If none of the input parameters were specified in the request, this data structure will contain the entire vocabulary.

Description:

The main purpose of the Agent vocabulary is to make the most common server specific information independent from a particular Agent implementation.

Example:

Retrieving the `counters` category parameters from the `generic` section of the vocabulary. This query will return the names of the performance counters used for server monitoring.

Input

```
<packet version="4.0.0">
  <data>
    <system>
      <get_vocabulary>
        <name>generic</name>
        <category>counters</category>
      </get_vocabulary>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/system"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="12c46713b39t5af1r81c" priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient3-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns2:vocabulary>
      <ns2:name>generic</ns2:name>
      <ns2:category>
        <id>counters_cpu</id>
        <category>generic</category>
        <category>counters</category>
        <short>CPU usage</short>
        <long>Hardware Node CPU related parameters</long>
      </ns2:category>
      <ns2:category>
        <id>counters_disk</id>
        <category>generic</category>
        <category>counters</category>
        <short>Disk usage</short>
        <long>Disk usage related parametres</long>
      </ns2:category>
      <ns2:category>
        <id>counters_memory</id>
        <category>generic</category>
        <category>counters</category>
        <short>Memory usage</short>
        <long>Memory usage related parametres</long>
      </ns2:category>
      <ns2:category>
        <id>counters_net</id>
        <category>generic</category>
        <category>counters</category>
```

```
        <short>Network usage</short>
        <long>Network usage related parametres</long>
      </ns2:category>
      <ns2:category>
        <id>counters_loadavg</id>
        <category>generic</category>
        <category>counters</category>
        <short>Load average</short>
        <long>CPU usage related parametres</long>
      </ns2:category>
      <ns2:category>
        <id>counters_process</id>
        <category>generic</category>
        <category>counters</category>
        <short>Process info</short>
        <long>Process info related parametres</long>
      </ns2:category>
      <ns2:category>
        <id>counters_system</id>
        <category>generic</category>
        <category>counters</category>
        <short>System info</short>
        <long>System info related parametres</long>
      </ns2:category>
    </ns2:vocabulary>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

login

Summary:

Logs the specified user in and creates a permanent session.

Request specification:

Name	Min/Max	Type	Description
`login`		`auth_nameType (p. 29)`	User login info. To get the list of the available realms, use the `system/get_realm` call (p. 545). You can execute the `get_realm` call without being logged in.
`{`			
`password`	`0..1`	`base64Binary`	User password.
`}`			

Returns:

Name	Min/Max	Type	Description
`token`	`1..1`	`tokenType`	A token containing the user security information.

Description:

The `login` call logs the specified user in and creates a permanent session. Once created, this type of session becomes the default session for the physical connection used. This means that if you execute an Agent request without specifying the session ID, the request will be sent through this session. A permanent session never expires. To close the permanent session, simply disconnect from Agent and the session will be automatically terminated.

Example:

Input

Logging in as the `root` user from the System realm (the host operating system user registry).

```
<packet version="4.0.0">
  <data>
    <system>
      <login>
        <name>cm9vdA==</name>
        <realm>00000000-0000-0000-0000-000000000000</realm>
        <password>bXlwYXNz</password>
      </login>
    </system>
  </data>
</packet>
```

Output

The output contains the security IDs of the user and the groups the user belongs to.

```
<ns1:packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/
system" xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="c9c46714aa5t135b8110r18fa" priority="0" version="4.0.0">
<ns1:origin>system</ns1:origin>
<ns1:target>vzclient14-cc98fba9-f1d6-fa46-b501-08dd4a0f0050</ns1:target>
<ns1:data>
  <ns1:system>
    <ns2:token xsi:type="ns3:tokenType">
      <ns3:user>AQUAAAAAIAGp+5jM1vFG+rUBCN1KDwBQAAAAAA==</ns3:user>
      <ns3:groups>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQAAAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQAQAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQCgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQAgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQAwAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQBAAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIACp+5jM1vFG+rUBCN1KDwBQBgAAAA==</ns3:sid>
        <ns3:sid>AQUAAAAAIAGp+5jM1vFG+rUBCN1KDwBQAAAAAA==</ns3:sid>
      </ns3:groups>
      <ns3:deny_only_sids/>
      <ns3:privileges/>
    </ns2:token>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

Input

Logging in as the `root` user of one of the Virtuozzo Containers. The Realm ID used here is the ID of the Virtuozzo Container Realm (one of the built-in Realms). When the Virtuozzo Container Realm ID is specified, it means that the authentication should be performed against the user registry inside the Virtuozzo Container specified in the `domain` field. The `domain` parameter contains the Server ID of the Container.

```
<packet version="4.0.0" id="3">
  <data>
    <system>
      <login>
        <name>cm9vdA==</name>
        <domain>ZTlhYjI4MzQtZWQ5Ny0xZjRiLWJkNDEtODFjMjdmYWNmYzMw</domain>
        <realm>00000000-0000-0000-0100-000000000000</realm>
        <password>TXlQYXNz</password>
      </login>
    </system>
  </data>
</packet>
```

ping

Summary:

Ping. Used to to test the availability of a host on a network.

Request specification:

Name	Min/Max	Type	Description
`ping`	`1..1`	`none`	

Returns:

OK/Error

Example 1:

Pinging the server that we are connected to.

Input

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <ping/>
    </system>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="15c467156a3t1ebr81c"
priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient3-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:system>
    <ns1:ok/>
  </ns1:system>
</ns1:data>
</ns1:packet>
```

Example 2:

This example pings a remote server over the exclusive connection that was established using the `connect` call (p. 525).

Input

```
<packet version="4.0.0">
  <target>192.168.0.841</target>
  <data>
    <system>
      <ping/>
    </system>
  </data>
</packet>
```

Output

In case of success, the call will return the standard Agent OK message. If the specified server cannot be found, a response similar to the following example will be returned:

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="1fc46715724t153cr81c" priority="0" version="4.0.0">
<ns1:origin>gend</ns1:origin>
<ns1:target>vzclient4-638a2a56-e689-c340-877d-bd0470f2448c</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
```

```
</ns1:dst>
<ns1:data>
  <ns1:error>
    <ns1:code>2</ns1:code>
    <ns1:message>Targets, specified in the message were not
found.</ns1:message>
  </ns1:error>
</ns1:data>
</ns1:packet>
```

register_client

Summary:

Registers a client with the Agent. Used with `count_registered` call (p. 527) to implement licensing functionality in the client software.

Request specification:

Name	Min/Max	Type	Description
`register_client`			
`{`			
`id`	`1..1`	`string`	An arbitrary string representing the client ID. The ID is used to identify the client software from which the connection has been initiated.
`}`			

Returns:

OK/Error

Description:

The `register_client` call and the `count_registered` call (p. 527) allow to keep track of the logged in clients and to limit the number of concurrent connections from the same client software to the Agent. You can use this functionality to implement a licensing policy where only a certain number of instances of your client software can be connected to the Agent at the same time. The following describes a typical usage scenario.

As soon as a client establishes a connection with the Agent, use the `count_registered` call (p. 527) to get the number of currently registered clients with the same ID. Depending on the result, one of the following should happen:

- If the number is less then the maximum allowed number of concurrent connections (the maximum number is determined by you) the login is granted. The new connection is then registered with the Agent using the `register_client` call thus incrementing the counter.
- If the number is equal to or greater than the maximum allowed number of connections, the login is denied and the connection is terminated.

It is not necessary to unregister the connection when the client logs off, as Agent does that automatically.

Please note that this call is used to count the permanent connections only. To count the user sessions, use the `sessionm/register_client` call (p. 480).

Example:

```
<packet version="4.0.0">
  <data>
    <system>
      <register_client>
        <id>My_Agent_Application</id>
      </register_client>
    </system>
  </data>
</packet>
```

subscribe

Summary:

Subscribes to system event notifications.

Request specification:

Name	Min/Max	Type	Description
`subscribe`			
`{`			
`name`	`1..[]`	`string`	Subscription name. See Events/Elements (p. 506) for the available subscriptions.

}			

Returns:

- OK/Error on initial execution.
- The following data structure when an event takes place:

Name	Min/Max	Type	Description
event	1..1	eventType **(p. 36)**	Event data.

Description:

Subscription is a mechanism allowing to monitor the system for critical events, such as the server status changes, Server configuration changes, and some others. It also allows to receive automatic notifications if an alert is raised on the server due to resource allocation problems. To subscribe to the event or an alert notification service, execute this call passing the subscription name.

As soon as the event takes place (or an alert is raised), a message is sent to your client program containing the event data. You recognize the message by examining the value of the `target` element in the message header, which should contain the same event subscription name as the one you used when you subscribed to the event notifications. Please remember that an Agent message may have more than one `target` element. When searching for a particular value, always remember to look through all occurrences of the `target` element.

A subscription remains in effect for the duration of the user session. If the client program disconnects and then re-connects again, the subscription is canceled and the client has to subscribe again. The events that triggered during that time (if any) will be unknown to this client. However, the majority of the events are logged internally by Agent, so you can retrieve this information later. The even log can be accessed using the `event_log` interface (p. 231).

For information on the available events and alerts, see the Events section (p. 504).

To stop receiving the event notifications, use the `unsubscribe` (p. 562) call.

Example:

Input

Subscribing to the environment status change events.

```
<packet version="4.0.0" id="2">
  <data>
    <system>
      <subscribe>
        <name>env_status_subscription</name>
      </subscribe>
    </system>
  </data>
</packet>
```

Output

Subscription was a success.

```
<packet priority="0" version="4.0.0" id="2">
  <origin>servd</origin>
  <target>vzclient1</target>
  <data>
    <system>
      <ok/>
    </system>
  </data>
</packet>
```

Output

The following is a notification message received when one of the Environments was stopped. The message contains the Server ID that generated the event, the text message describing the event, and the event data (the old and the new server state and transition codes). Note that one of the `target` elements contains the same value as the one we used in the `name` element in the request, which is `env_status_subscription` -- that's how you recognize the message. Please also note that the inner `data` structure contains the elements specific to this event type, which in this particular case is the `env_status_event` element (p. 506).

```
<packet version="4.0.0">
  <target>events_subscription</target>
  <target>env_status_subscription</target>
  <data>
    <event>
      <eid>849c9be9-5fbb-4e7d-b100-f841f86c150e</eid>
      <time>1155317636</time>
      <source></source>
      <category>env_status_subscription</category>
      <sid>XXX</sid>
      <data>
        <env_status_event>
          <eid>62ec514e-bc38-4aee-830d-cc802ee2aadd</eid>
          <new>
            <state>3</state>
          </new>
          <old>
            <state>3</state>
            <transition>5</transition>
          </old>
        </env_status_event>
      </data>
      <info>
        <message>

RW52aXJvbm1lbnQgJWVpZCUgc3RhdHVzIGNoYW5nZWQgZnJvbSAlb2xkJSB0byAlbmV3JQ==
        </message>
        <name></name>
        <translate/>
        <parameter>
          <message>NjJlYzUxNGUtYmMzOC00YWVlLTgzMGQtY2M4MDJlZTJhYWRk</message>
           <name>eid</name>
        </parameter>
        <parameter>
          <message>Mw==</message>
          <name>new</name>
          <translate/>
        </parameter>
        <parameter>
          <message>Mw==</message>
          <name>old</name>
          <translate/>
        </parameter>
      </info>
    </event>
  </data>
  <src>
    <director>gend</director>
  </src>
</packet>
```

uninstall

Summary:

Uninstalls Agent from the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
`uninstall`			
`{`			
`shutdown`	`0..1`	`none`	If this element is included, the Hardware Node will be automatically shut down at the end of the Agent uninstall operation.
`}`			

Returns:

OK/Error

unsubscribe

Summary:

Cancels the specified event subscription. This call is the opposite of the `subscribe` call (p. 558).

Request specification:

Name	Min/Max	Type	Description
`unsubscribe`			
`{`			
`name`	`1..[]`	`string`	The names of the subscriptions that you would like to cancel.
`}`			

Returns:

OK/Error

The progress packet

Purpose:

This is a special packet that is used to report progress information during long operations.

Output specification:

Name	Min/Max	Type	Description
`progress`		`progress_eventType (p. 565)`	Progress data.

Description:

Only certain types of operations are able to produce progress information. The examples of such operations include creating a Virtuozzo Container, backing up a Container, and others. By default, the progress information is not sent to the client by Agent. In order to receive the information, you must include the `progress="on"` and the `id` attributes in the `packet` element of the message header. Optionally, you can also include the `log="on"` attribute, which will initiate logging of the progress data in the history database.

Example:

The following example shows how to start a Virtuozzo Container backup operation and turn the progress reporting on.

Input

```
<packet progress="on" log="on" id="2" version="4.0.0">
  <target>backupm</target>
  <data>
    <backupm>
      <backup_env>
        <env_list>
          <eid>57c2cd6c-c02b-4645-bdb5-e883ea005896</eid>
        </env_list>
      </backup_env>
    </backupm>
  </data>
</packet>
```

Progress packet:

The following is an example of one of the progress packets received by the client program in response to the request listed above.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="7c45db0eb2t72aer488"
time="2007-02-19T15:01:32+0000" type="1" priority="4000" version="4.0.0">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns1:progress>
    <ns1:op>backupm.backup_env</ns1:op>
    <ns1:message>
      <ns1:message>Operation %op_name% is %status%</ns1:message>
      <ns1:name></ns1:name>
      <ns1:translate/>
      <ns1:parameter>
        <ns1:message>YmFja3VwX2Vudg==</ns1:message>
        <ns1:name>op_name</ns1:name>
      </ns1:parameter>
      <ns1:parameter>
        <ns1:message>c3RhcnRlZA==</ns1:message>
        <ns1:name>status</ns1:name>
        <ns1:translate/>
      </ns1:parameter>
    </ns1:message>
    <ns1:status>1</ns1:status>
  </ns1:progress>
</ns1:data>
```

```
<ns1:target>log_subscription</ns1:target>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

The packet above needs to be decoded. See `infoType (p. 38)` for the information on how it is done. Once you parse the packed, resolve all the parameters and their values, and decode the base64-encoded values, the actual message will look like this:

```
Operation backup_env is started.
```

Here's a different example of a progress reporting packet. In this case, the packet does not contain any text message but provides the operation status and the progress percentage.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="7c45db0eb2t72aer488"
time="2007-02-19T15:03:25+0000" type="1" priority="4000" version="4.0.0">
<ns1:origin>backupm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns1:progress>
    <ns1:percent>20</ns1:percent>
    <ns1:status>2</ns1:status>
  </ns1:progress>
</ns1:data>
<ns1:target>log_subscription</ns1:target>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Types

progress_eventType

Summary:

Progress information.

Type specification:

Name	Min/Max	Type	Description
`message`	`0..1`	`infoType (p. 38)`	Progress message. Usually contains a text describing the stage of the operation and may also contain parameters describing the Agent operators involved and some other values.
`percent`	`0..1`	`int`	Progress percentage.
`eid_list`	`0..1`	`eid_listType (p. 32)`	The list of Environments participated in the operation.

op	0..1	string	The name of the operation. This is usually the name of the call that initiated the operation. For example, backupm.backup_env.
status	0..1	int	Operation status: 1 -- started. 2 -- processing. 3 -- success. 4 -- error.
id	0..1	int	The original packet ID. This is the ID that you assigned to the original request. Use this ID to match the request and the response messages.
time	0..1	datetime_type	The time elapsed since the operation was started.
parent_id	0..1		Certain types of operations are initiated automatically by other operations. For such operations, this element will contain the name of a parent operation.

Description:

For more information on progress reporting, see The progress packet section (p. 563).

CHAPTER 4

Virtuozzo Containers Types and Interfaces

This chapter describes the types and interfaces that are specific to the Virtuozzo Containers management only. The majority of the types and interfaces described here are derived by extension from the base types and interfaces (p. 23).

In This Chapter

Common Types 568
Interfaces 579

Common Types

This chapter describes the common data types. There are two main categories of common types:

- Simple Types are custom types that contain the actual data.
- Complex Types are custom types that can contain data, attributes and other elements.

Each category is described in detail in the following sections.

Simple Types

Simple types are custom types that can contain a value but cannot contain attributes or elements. Most of the custom simple types have restrictions added to them in order to limit their content. A restriction can limit the type to a specific primitive data type, it can also define a list of enumerated values, or it can define a string pattern that the value must adhere to.

veid_type

Summary:

Virtuozzo Container ID. The ID is assigned to a Container by the user when the Container is created. This ID is different from the Server ID, which is a globally unique ID assigned to every server (physical or virtual) automatically by Agent.

Type specification:

Restriction: `long`

Complex Types

Complex types are custom types that can contain text, attributes and other elements. This section describes the complex types specific to the Virtuozzo Containers portion of the Parallels Agent API.

log_optionsType

Summary:

Log retrieval options.

Type specification:

Extends `log_options_baseType (p. 45)`

Adds the following elements:

Name	Min/Max	Type	Description
type	0..1	int	Log type.

net_vethType

Summary:

Contains Virtuozzo virtual network adapter configuration information.

Type specification:

Extends net_nicType (p. 49)

Adds the following elements:

Name	Min/Max	Type	Description
wins_server	0..[]	string	A list of WINS servers.
nameserver	0..[]	string	A list of name servers.
default_gateway	0..1	string	Default gateway info (hostname or IP address).

host_routed	0..1	none	If present, indicates that the adapter is using the host-routed communication mode. On Windows, Virtuozzo virtual ethernet adapters inside a Container can operate in both the host-routed and the bridged modes. Include this element if you want the virtual adapter to use the host-routed mode. To use the bridged mode, omit the element and specify the Virtuozzo virtual network ID using the network_id element (the network ID identifies the physical network adapter on the Hardware Node). On Linux, the default venet0 adapter can operate in the host-routed mode only, while custom virtual network adapters (the adapters that you create manually) can operate only in the bridged mode. When configuring the venet0 adapter (setting the IP address for example), you must include this element in the request. The adapter name (venet0) must also be specified using the id element.

The following examples illustrate various Virtuozzo Container network adapter configuration scenarios.

Example 1:

Specifying the IP address for the default `venet0` virtual ethernet adapter inside a Container.

```
<id>venet0</id>
<ip_address>
  <ip>10.17.3.125</ip>
</ip_address>
<host_routed/>
```

Example 2:

Creating a virtual ethernet adapter inside a Container. Naming the adapter `veth1` and connecting it to the `vznet5` Virtuozzo virtual network (for more information on Virtuozzo virtual network and its components see `vzanetworkm` (p. 606)).

```
<id>veth1</id>
<ip_address>
  <ip>10.17.3.126</ip>
</ip_address>
<network_id>vznet5</network_id>
```

Example 3:

Specifying the IP address for the default `venet0` virtual ethernet adapter inside a Container and switching the adapter's operation mode to `host-routed`.

```
<id>venet0</id>
<ip_address>
  <ip>10.17.3.121</ip>
</ip_address>
<host_routed/>
```

Example 4:

Switching the operation mode of the `venet0` adapter to `bridged` by connecting it to the `vzanet3` Virtuozzo virtual network. The network ID on Windows identifies a physical network adapter or a non-Virtuozzo virtual network. The ID must be created using the `vzanetworkm/set` (p. 307) call before you can use it here.

```
<id>venet0</id>
<network_id>dnpuZXQz</network_id>
```

package_std_vztemplateType

Summary:

Contains information about a standard Virtuozzo template. Use this type when installing a standard Virtuozzo template.

Type specification:

Extends `package_vztemplateType (p. 572)`

Adds the following elements:

Name	Min/Max	Type	Description
`base`	`0..1`	`boolean`	`true` indicates that this is a base (standalone) version; `false` indicates otherwise.

package_vztemplateType

Summary:

Contains information about a Virtuozzo EZ template.

Type specification:

Extends `packageType (p. 50)`

Adds the following elements:

Name	Min/Max	Type	Description
`technology`	`0..[]`	`string`	Required or supported technologies.
`os_template`	`1..1`	`boolean`	`true` indicates that this is an OS template; `false` indicates otherwise. When using the `get_info` call (p. 343) to obtain information about a template, this element must be included in the request and must indicate the template type.
`cached`	`1..1`	`boolean`	`true` indicates that the package was cached; false indicates otherwise.
`path`	`0..1`	`base64Binary`	The location of the package.

uptodate	1..1	boolean	true indicates that the cached OS template is up to date; false indicates otherwise.

Subtypes:

package_std_vztemplateType (p. 572)

redirect_serviceType

Summary:

Sets the offline redirect information. Used by the Virtuozzo Container offline management feature.

Type specification:

Name	Min/Max	Type	Description
id	1..1	string	Redirect ID.
port	1..1	int	Redirect port.
dst	1..1	eid_type (p. 25)	Redirect Server ID.
default	0..1	none	If set, this service is added to the list of the default services.

Description:

Sets the offline redirect information: the service ID (vzpp or vzpp-plesk), the active port reserved for the offline redirection service, the ID of the Container to which the requests coming to the specified port will be redirected.

templateType

Summary:

Application template name and version.

Type specification:

Name	Min/Max	Type	Description
name	1..1	string	Template name.
version	0..1	string	Template version.

venv_configType

Summary:

Virtuozzo Container configuration.

Type specification:

Extends `venv_configType` (p. 61)

Adds the following elements:

Name	Min/Max	Type	Description
`veid`	`0..1`	`vzat:veid_type` (p. 568)	Virtuozzo Container ID.
`ve_root`	`0..1`	`string`	Root directory.
`ve_private`	`0..1`	`string`	Private area directory.
`on_boot`	`0..1`	`boolean`	A flag indicating whether the Container should be started on system boot: `true` -- start Container on system boot. `false` -- do not start Container.
`template`	`0..[]`	`templateType` (p. 573)	A list of application templates used by the Container.
`disabled`	`0..1`	`boolean`	A flag indicating whether the Container is enabled or disabled: `true` -- the Container is disabled. `false` -- the Container is enabled.
`offline_management`	`0..1`	`boolean`	A flag indicating whether the offline management is enabled for the Container: true -- the offline management is enabled. false -- the offline management is disabled.
`os_template`	`0..1`	`templateType` (p. 573)	The name of the OS template used by the Container.

rate_bound	0..1	boolean	Activates network interface limitations. Applicable only if network traffic shaping is turned on. true -- use the per-Container network rate defined for a network interface. The Container will not be allowed to exceed this limit and to borrow additional traffic from the total rate defined for an interface. false -- do not limit network traffic for this Container individually. The traffic will still be limited by the total rate defined for an interface.
distribution	0..1	templateType (p. 573)	Linux distribution information.
capability	0..[]		A list of capabilities.
{			
id	1..1	string	Capability ID.
value	1..1	boolean	Value.
}			
iptables	0..[]	string	A list of IP table modules.
config_customized	0..1	boolean	Indicates whether the configuration of the Container was modified after the Container was created: true -- the original configuration was modified. false -- the original configuration is intact.
class_id	0..1	string	*The parameter is obsolete and is left for compatibility purposes only.*
ve_type	0..1		Indicates whether the Container is a repair Container -- a substitute for another Container being repaired.
{			
veid	0..1	veid_type (p. 568)	The ID of a Container being repaired that this Container is substituting for.

type	1..1	int	The Container type. If the ve_type element (and consequently this element) is present, the value is always repair.
}			
offline_service	0..[]	string	A list of offline services. Currently supported services are vzpp and vzpp-plesk.
wins_server	0..[]	string	IP address list for WINS configuration.
net_device	0..[]	net_vethType (p. 569)	Network settings. on Linux, the element is used to specify the parameters that will be used to create or to configure virtual ethernet adapters inside the Container. It is also used to configure the default venet0 virtual adapter. In Windows, the parameter is used to configure the default venet0 virtual adapter.
ts_license_server	0..[]	string	Terminal server (TS) license server list.
ts_mode	0..1	int	Terminal server (TS) mode: 0 -- APPLICATION 1 -- ADMIN
uuid	0..1	string	An internal unique server ID. Used for local installation of EZ templates and by vzcache command-line utility.
allow_reboot	0..1	boolean	Allow Container rebooting.
rate_bound	0..1	boolean	If set to true, traffic shaping will be turned on for this Container. See set_config (p. 608) for more info on traffic shaping.
interface_rate	0..[]		Traffic shaping information.
{			
class_id	1..1	string	VIrtuozzo network class ID. See vzanetworkm (p. 606) for more info on network classes.

rate	1..1	long	Traffic rate in Kbps. This value overrides (for this Container only) the rate specified for the class on the Hardware Node level. See vzanetworkm (p. 606) for more info on network classes.
slm_mode	0..1	string	Memory management mode: slm -- Service Level Management. ubc -- old-style UBC memory management. all -- both SLM and UBC memory management.
}			

virtuozzo_configType

Summary:

Contains the Virtuozzo Container configuration data in bash style -- the format used by Virtuozzo to store configurations.

Type specification:

Extends native_configType (p. 47)

Adds the following elements:

Name	Min/Max	Type	Description
body	1..1	base64Binary	The Container configuration data in bash style: VARNAME="value-string"

vt_infoType

Summary:

Read-only Virtuozzo Containers information.

Type specification:

Extends `vzlt:vt_infoType (p. 63)`

Adds the following elements:

Name	Min/Max	Type	Description
`sve_eid`	`1..1`	`eid_type (p. 25)`	Service Container ID.
`version`	`1..1`	`string`	Virtuozzo Containers version.
`release`	`1..1`	`string`	Virtuozzo Containers release.

vt_settingsType

Summary:

Contains global Virtuozzo Containers settings.

Type specification:

Extends `vt_settingsType (p. 63)`

Adds the following elements:

Name	Min/Max	Type	Description
`parameter`	`0..[]`		List of parameters.
`{`			
`id`	`1..1`	`string`	Parameter ID.
`value`	`1..1`	`string`	Parameter Value.
`}`			
`service`	`0..[]`	`redirect_serviceType (p. 573)`	Sets offline redirect information.
`qos`	`0..[]`	`qosType (p. 53)`	QoS parameters.

Interfaces

The material in this section describes Virtuozzo Containers API interfaces. The term *interface*, as we use it, is somewhat similar to a class in object-oriented programming. We use interfaces to group related data types (structures) and calls (methods). The data types and calls are defined using XML Schema language (XSD). The body of an Agent XML request always begins with the name of an interface followed by the name of a call. The rest of the request body is composed according to the call specifications.

The interfaces described in this chapter provide Virtuozzo Containers management functionality. Please note that the interfaces described here do not comprise a complete list of the Virtuozzo Containers functions. For the complete list please also see the Base Types and Interfaces chapter (p. 23).

vzadevm

Purpose:

The device management interface.

Specification:

Derived from the `devm` interface (p. 160).

Calls

Call	Description
get_mounts (p. 172)	Retrieves a list of devices mounted in the specified Container.
new_mount (p. 180)	Mounts the specified device.
umount (p. 186)	Unmounts the specified device.
get_info (p. 171)	Gets information about all available filesystems, partitions and devices.
create_drive (p. 163)	Creates a new file system image file and loopback-mounts it in the specified Container.
delete_drive (p. 167)	Unmounts and deletes the file system image file.
resize_drive (p. 185)	Resizes file system image file.
format_drive (p. 168)	Formats a disk drive.
list_devices (p. 176)	Lists devices available on the Hardware Node.
forward_device (p. 169)	Makes a SCSI device on the Hardware Node visible and accessible to Virtuozzo Containers.
remove_forward	Cancels forwarding of a device (see `forward_device` above).

list_forward (p. 178)	List the device forwarding information (see `forward_device` above).

vzaenvm

Purpose:

Virtuozzo Container management interface.

Specification:

Derived from the `envm` interface (p. 193)

Types

ugid_quota_info

Summary:

Second-level disk quota info.

Type specification:

Name	Min/Max	Type	Description
`type`	`1..1`	`int`	Specifies whether this quota is for a user or a group: `0` -- User `1` -- Group
`quota`	`0..[]`	`quota_type (p. 581)`	Quota values.
`grace_period`	`0..1`		Grace periods for second-level quotas (affects all other Container quotas).
`{`			
`  inodes`	`1..1`	`int`	Disk inodes grace period in seconds.
`  space`	`1..1`	`int`	Disk space grace period in seconds.
`}`			

quota_type

Summary:

Second-level disk quota values.

Type specification:

Name	Min/Max	Type	Description
`id`	`1..1`	`int`	User or group ID.
`diskspace`	`1..1`	`quota_limit` (p. 581)	Disk space limitations.
`diskinodes`	`1..1`	`quota_limit` (p. 581)	Disk inodes limitations.

quota_limit

Summary:

Second-level disk quota limits.

Type specification:

Name	Min/Max	Type	Description
`cur`	`1..1`	`long`	Current value.
`soft`	`1..1`	`long`	Soft limit.
`hard`	`1..1`	`long`	Hard limit.

validationType

Summary:

Contains the results of the Virtuozzo Container configuration validation.

Type specification:

Name	Min/Max	Type	Description
`type`	`1..1`	`int`	Warning level from 0 to 3 with 0 being the most critical error.
`warning`	`1..1`	`string`	A warning message.

formula	1..1	string	A mathematical statement expressing the Container configuration rule that has been violated. The statement consists of names of the QoS counters as variables, and the usual mathematical symbols. Please note that the < (less than) and > (greater than) signs will be substituted with < and > respectively. Consider the following example: tcpsndbuf.hard - tcpsndbuf.soft >= 2.5kb * numtcpsock By substituting the > sequence in the statement above, we'll get the formula: tcpsndbuf.hard - tcpsndbuf.soft >= 2.5kb * numtcpsock
qosID	1..[]	string	The IDs of the QoS counters that violated the rule specified in the formula element.

Calls

Call	Description
create (p. 195)	Creates a new Virtuozzo Container.
repair (p. 215)	Creates a Virtuozzo Container as a temporary replacement for another Container that needs repairs.
stop_repair (p. 226)	Stops and destroys the temporary Container created by the repair call.
start (p. 224)	Starts the specified Container.
stop (p. 225)	Stops the specified Container.
restart (p. 215)	Restarts a Container..
destroy (p. 201)	Destroys an Container.
get_info (p. 202)	Retrieves information about a Container.
get_list (p. 208)	Gets a list of Virtuozzo Containers.
set (p. 217)	Sets the Container configuration parameters.
get_vt_settings (p. 212)	Retrieves Virtuozzo Containers settings.
set_vt_settings (p. 223)	Allows to modify Virtuozzo Containers settings.
get_vt_info (p. 211)	Retrieves read-only Virtuozzo Containers information.

get_native_config (p. 228)	Obtains a native Container configuration based on the provided virtual configuration.
get_virtual_config	Obtains virtual configuration based on the provided native Container configuration..
mount (p. 590)	Mounts a Container private area.
umount (p. 598)	Unmounts a Container private area.
suspend (p. 597)	Suspends a Container.
resume (p. 593)	Resume a Container.
set_user_password (p. 596)	Sets user password in a Container.
upgrade (p. 599)	Upgrades a Container.
determine_env (p. 584)	Gets Server ID of a Container by IP address. *Internal call. Not used in client applications.*
set_ugid_quota (p. 595)	Sets second-level disk quota.
get_ugid_quota (p. 589)	Gets second-level disk quota.
get_split_conf (p. 586)	Calculates an optimal sample Container configuration.
validate (p. 600)	Validates a sample Container configuration.
get_script (p. 585)	Retrieves a list of scripts used during reinstallation of a Virtuozzo Container.
set_script (p. 594)	Adds a custom script to a Virtuozzo Container that can later be used while reinstalling the Container.
del_script (p. 583)	Deletes a script from a Container.
recover_template (p. 591)	Attempts to recover all Virtuozzo Container templates.

del_script

Summary:

Adds a re-installation script to the Container.

Request specification:

Name	Min/Max	Type	Description
`del_script`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	The ID of the Container to delete the script from.
`type`	`0..1`	`string`	Script type. The only available type at the time of this writing is `reinstall`.
`name`	`0..1`	`string`	The name of the script to delete.
`}`			

Returns:

OK/Error

Description:

The `del_script` call allows to delete a reinstall script from the Container. For more information on reinstall scripts see the `get_script` (p. 585) and `set_script` (p. 594) calls.

determine_env

Summary:

Determines the Server ID of a Virtuozzo Container by the specified IP information.

This call is used by Virtuozzo tools only and should not be used in client programs.

Request specification:

Name	Min/Max	Type	Description
`determine_env`			
`{`			
`link`	`1..1`		
`{`			
`ip`	`1..1`	`ip_type` (p. 26)	Container IP address.
`port`	`0..1`	`int`	Container port number.
`client_ip`	`0..1`	`ip_type` (p. 26)	Client IP address.
`client_port`	`0..1`	`int`	Client IP port number.
`}`			
`}`			

Returns:

Name	Min/Max	Type	Description
`eid`	`0..1`	`eid_type` (p. 25)	Server ID.

get_script

Summary:

Retrieves a list of scripts used during reinstallation of a Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
`get_script`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	The ID of the server to get the scripts for.
`type`	`0..1`	`string`	Script type. The only available type at the time of this writing is `reinstall`.
`name`	`0..1`	`string`	If you would like to retrieve the information about a particular script, specify its name here. Omit the element to retrieve the information about all available scripts.
`}`			

Returns:

Name	Min/Max	Type	Description
`script`	`1..1`		
`{`			
`name`	`1..1`	`string`	The name of the scripts.
`type`	`1..1`	`string`	Script type.
`description`	`0..1`	`base64Binary`	Script description.
`body`	`0..1`	`base64Binary`	The body of the script.
`}`			

Description:

A Virtuozzo Container can be reinstalled if the administrator has inadvertently modified, replaced, or deleted any of the files that are a part of an application or operating system template thus breaking the Container itself or an application installed inside it. During reinstallation, a series of scripts is executed in order to bring the application templates inside a Container to their original working states. Every Container may have its own set of scripts depending on the application templates installed in it. The `get_script` call allows to retrieve a list of the available scripts. The list can then be used to select the scripts to execute if you are performing custom reinstall. See the `vzatbs/fix` call for more information.

To add a custom script to a Container, use the `set_script` call (p. 594).

get_split_conf

Summary:

Calculates an optimal sample Container configuration.

Request specification:

Name	Min/Max	Type	Description
`get_split_conf`	`1..1`		
`{`			
`number`	`1..1`		The projected number of Containers.
`}`			

Returns:

Name	Min/Max	Type	Description
`config`	`1..1`	`vzat:venv_configType (p. 574)`	The calculated sample configuration.

Description:

If you know that a given Hardware Node will be hosting a certain maximum number of Containers, you can use the `get_split_conf` call to calculate an optimal sample Container configuration, which can be used to create Containers. The calculation is done based on the available Hardware Node resources by dividing them equally between the projected number of Containers. Before creating Containers using the sample configuration produced by this call, it is important to validate the configuration with the `validate` call (p. 600).

The `get_split_conf` call utilizes a Virtuozzo command-line tool called `vzsplit`. For more information on `vzsplit` please refer to Virtuozzo documentation.

Example:

Getting the sample configuration for the total of 20 Virtuozzo Containers.

Input

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_split_conf>
        <number>20</number>
      </get_split_conf>
    </vzaenvm>
  </data>
</packet>
```

Output

```
<ns1:packet priority="0" version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:env_config xsi:type="ns3:venv_configType">
      <ns3:qos>
        <ns3:id>avnumproc</ns3:id>
        <ns3:hard>96</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>cpuunits</ns3:id>
        <ns3:hard>6677</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>dcachesize</ns3:id>
        <ns3:hard>1769472</ns3:hard>
        <ns3:soft>1717933</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>dgramrcvbuf</ns3:id>
        <ns3:hard>497780</ns3:hard>
        <ns3:soft>497780</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>diskinodes</ns3:id>
        <ns3:hard>88000</ns3:hard>
        <ns3:soft>80000</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>diskspace</ns3:id>
        <ns3:hard>225280</ns3:hard>
```

```
        <ns3:soft>204799</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>kmemsize</ns3:id>
        <ns3:hard>8692068</ns3:hard>
        <ns3:soft>7901880</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>lockedpages</ns3:id>
        <ns3:hard>385</ns3:hard>
        <ns3:soft>385</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numfile</ns3:id>
        <ns3:hard>3072</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numflock</ns3:id>
        <ns3:hard>337</ns3:hard>
        <ns3:soft>307</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numiptent</ns3:id>
        <ns3:hard>100</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numothersock</ns3:id>
        <ns3:hard>400</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numproc</ns3:id>
        <ns3:hard>400</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numpty</ns3:id>
        <ns3:hard>40</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numsiginfo</ns3:id>
        <ns3:hard>1024</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>numtcpsock</ns3:id>
        <ns3:hard>400</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>oomguarpages</ns3:id>
        <ns3:hard>2147483647</ns3:hard>
        <ns3:soft>11701</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>othersockbuf</ns3:id>
        <ns3:hard>2136180</ns3:hard>
        <ns3:soft>497780</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>physpages</ns3:id>
        <ns3:hard>2147483647</ns3:hard>
        <ns3:soft>0</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>privvmpages</ns3:id>
        <ns3:hard>12871</ns3:hard>
        <ns3:soft>11701</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>shmpages</ns3:id>
```

```
        <ns3:hard>1170</ns3:hard>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>tcprcvbuf</ns3:id>
        <ns3:hard>2633960</ns3:hard>
        <ns3:soft>995560</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>tcpsndbuf</ns3:id>
        <ns3:hard>2633960</ns3:hard>
        <ns3:soft>995560</ns3:soft>
      </ns3:qos>
      <ns3:qos>
        <ns3:id>vmguarpages</ns3:id>
        <ns3:hard>2147483647</ns3:hard>
        <ns3:soft>11701</ns3:soft>
      </ns3:qos>
    </ns2:env_config>
  </ns2:vzaenvm>
</ns1:data>
</ns1:packet>
```

get_ugid_quota

Summary:

Gets second-level disk quota.

Request specification:

Name	Min/Max	Type	Description
`get_ugid_quota`	`1..1`		
`{`			
`eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`id`	`0..[]`	`int`	User or group ID to get the quota for.
`type`	`1..1`	`int`	Specifies whether the `id` element contains a user ID or a group ID: `0` -- User. `1` -- Group.
`}`			

Returns:

Name	Min/Max	Type	Description
`ugid_quota`	`1..1`	`ugid_quota_info (p. 580)`	The requested second-level quota info.

Description:

Virtuozzo supports two levels of disk quotas that can be set for a Container: *first-level quotas* and *second-level quotas*.

First-level quotas enable system administrator to limit the amount of disk space and the number of inodes that an individual Container can use. First-level quotas are enabled in Virtuozzo by default but can be turned on or off for an individual Container through Container configuration.

Second-level quotas allow to limit the amount of disk space and the number of inodes that an individual user or a group in a Container is allowed to use.

Example:

Getting second-level disk quota for the user "root" in the specified Container.

Input:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <get_ugid_quota>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
        <id>0</id>
        <type>0</type>
      </get_ugid_quota>
    </vzaenvm>
  </data>
</packet>
```

mount

Summary:

Mounts a Container private area.

This is a logged operation.

Request specification:

Name	Min/Max	Type	Description
mount			
{			
eid	1..1	eid_type (p. 25)	Server ID.
}			

Returns:

OK/Error

Description:

The `mount` call allows to mount a Container private area without actually starting the Container itself. Once the private area is mounted, it can be accessed at the `/vz/root/[veid]` path on the Hardware Node. To unmount a Container, use the `umount` call (p. 186).

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <mount>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
      </mount>
    </vzaenvm>
  </data>
</packet>
```

recover_template

Summary:

The `recover_template` call is used to restore the original state of a Virtuozzo Container system and application files in case the Container administrator has inadvertently modified, replaced, or deleted any of the files (more precisely the VZFS symlinks in the Container private area) that are part of an application or operating system template. The symlinks may be restored by rewriting the original symlinks to the Container private area (recover) or by creating an entirely new private area to replace the existing one (reinstall).

Request specification:

Name	Min/Max	Type	Description
`recover_template`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	Server ID of the Container.
`password`	`0..1`	`base64Binary`	Use this element to specify the new password for the user `root`.

clean	0..1	none	Include this element to reinstall the Container private area. The procedure consists of the following steps: 1. A new private area is created. 2. All installed applications are reinstalled in the new private area. 3. If the password option (above) is NOT included, all existing user profiles will be copied to the new private area. If the option is included, the users will be discarded. 4. If the skipbackup option (below) is NOT included, the old Container private area will be backed up to the /old directory on the host machine. Note: If you use non-default configurations on some of the services, the services will be available in the /old directory only. Thus, some of the services may not work as intended.
skipbackup	0..1	none	By default, the old private area of the Container is backed up to the /old directory on the Hardware Node. Include this element if you don't want to create a backup.
script	0..[]	string	During the reinstallation procedure, a series of scripts is executed in order to bring the application templates inside a Container to their original working states. These scripts are supplied by the application vendors and are copied to the Hardware Node when you initially install an application template. By default all available scripts will be executed automatically. You may use this element to manually specify the names of the scripts to execute. To retrieve the names of the available scripts use the get_script call (p. 585).

}			

Returns:

OK/Error

Example:

The following example demonstrates how to recover the Container private are using the default options.

Input

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <recover_template>
        <eid>6576a097-9823-ea48-94a1-4c09e824292c</eid>
      </recover_template>
    </vzaenvm>
  </data>
</packet>
```

The following example demonstrates how to reinstall the Container private area.

Input

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <recover_template>
        <eid>6576a097-9823-ea48-94a1-4c09e824292c</eid>
        <password>bmV3cGFzcw==</password>
        <clean/>
      </recover_template>
    </vzaenvm>
  </data>
</packet>
```

resume

Summary:

Resumes a suspended Container.

Request specification:

Name	Min/Max	Type	Description
resume			
{			
eid	1..1	eid_type (p. 25)	Server ID.
}			

Returns:

OK/Error

Description:

Resumes the normal operation of a Container that was previously suspended with the `suspend` call (p. 597).

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <resume>
        <eid>3e25fee2-1163-4336-9e74-8b8097936d47</eid>
      </resume>
    </vzaenvm>
  </data>
</packet>
```

set_script

Summary:

Adds a re-installation script to the Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
`set_script`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	The ID of the Container to add the script to.
`type`	`0..1`	`string`	Script type. The only available type at the time of this writing is `reinstall`.
`name`	`0..1`	`string`	A name that you would like to use for he script.
`body`	`0..1`	`base64Binary`	The body of the script.
`}`			

Returns:

OK/Error

Description:

A Virtuozzo Container can be reinstalled if the administrator has inadvertently modified, replaced, or deleted any of the files that are a part of an application or operating system template thus breaking the Container or an application installed inside it. During reinstallation, a series of scripts are executed in order to bring the application templates inside a Container to their original working states. Every Container may have its own set of scripts depending on the application templates installed in it. The `set_script` call allows to add a custom script to the Container that can be used later while performing a custom Container reinstall. See the `vzatbs/fix` call for more information.

To get the list of the available scripts, use the `get_script` call (p. 585).

set_ugid_quota

Summary:

Sets second-level disk quota.

Request specification:

Name	Min/Max	Type	Description
`set_ugid_quota`	`1..1`		
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	Server ID.
`ugid_quota`	`1..1`	`ugid_quota_info` (p. 580)	Second-level quota information.
`}`			

Returns:

OK/Error

Description:

Virtuozzo supports two levels of disk quotas that can be set for a Container: *first-level quotas* and *second-level quotas*.

First-level quotas enable system administrator to limit the amount of disk space and the number of inodes that an individual Container can use. First-level quotas are enabled in Virtuozzo by default but can be turned on or off for an individual Container through Container configuration.

Second-level quotas allow to limit the amount of disk space and the number of inodes that an individual user or a group in a Container is allowed to use. Second-level quotas are not turned on by default and must be set for each Container separately. To set second-level quotas for a Container, first-level quotas must be turned on for that Container.

Example:

Setting second-level quotas for the user "root" in the specified Container.

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set_ugid_quota>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
        <ugid_quota>
           <type>0</type>
           <quota>
             <id>0</id>
             <diskspace>
               <soft>1000000</soft>
               <hard>1500000</hard>
             </diskspace>
             <diskinodes>
               <soft>1000</soft>
               <hard>15000</hard>
             </diskinodes>
           </quota>
         </ugid_quota>
       </set_ugid_quota>
    </vzaenvm>
  </data>
</packet>
```

set_user_password

Summary:

Sets a user password in a Container, creating the user if he/she doesn't exist.

Request specification:

Name	Min/Max	Type	Description
set_user_password			

{			
eid	1..1	eid_type (p. 25)	Server ID.
name	0..1	string	User name. If not specified, the default user with administrative privileges is used.
password	1..1	base64Binary	New password.
}			

Returns:

OK/Error

Description:

To set a user password, a Container doesn't have to be running. If it is stopped, the Container private area will be automatically mounted and then unmounted on completion. If the user doesn't exist, a new user will be added to the user registry using the supplied name and password.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <set_user_password>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
        <name>johndoe</name>
        <password>bXlwYXNz</password>
      </set_user_password>
    </vzaenvm>
  </data>
</packet>
```

suspend

Summary:

Suspends a running Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
suspend			
{			
eid	1..1	eid_type (p. 25)	Server ID.
}			

Returns:

OK/Error

Description:

Use this call to suspend a running Container without shutting it down. You can suspend a Container at any time. If a Container temporarily cannot be suspended (it is in a transitional status for example), the call will return an error. The status of a suspended Container is "suspended". To a resume the Container operation, use the `resume` call (p. 593).

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <suspend>
        <eid>78c510e9-6e4f-5349-9e22-48841e709fea</eid>
      </suspend>
    </vzaenvm>
  </data>
</packet>
```

umount

Summary:

Unmounts a Container private area.

Request specification:

Name	Min/Max	Type	Description
`unmount`			
`{`			
`  eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`}`			

Returns:

OK/Error

Description:

Un-mounts a Container private area that was previously mounted using the `mount` call (p. 590). You cannot unmount private area of a Container that is running (the `stop` call (p. 225) unmounts the Container private area automatically.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <umount>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
      </umount>
    </vzaenvm>
  </data>
</packet>
```

upgrade

Summary:

Upgrades the OS template used by the specified Container.

Request specification:

Name	Min/Max	Type	Description
`upgrade`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	Server ID.
`options`	`0..1`		Upgrade options.
`{`			
`force`	`0..1`	`none`	Force the operation if possible.
`}`			
`}`			

Returns:

OK/Error

Description:

The upgrade call can perform two kinds of upgrades depending on the OS template that the specified Container is currently using:

1. If the Container uses an old-style Virtuozzo OS template, the call will search for the equivalent EZ template on the Hardware Node. If the template is available, the Container will be upgraded to use that template. If the appropriate EZ template is not available, the call will return an error.
2. If the Container uses an EZ template already, the call will check if a newer version is installed on the Hardware Node, and will upgrade the Container if there's.

Example:

```
<packet version="4.0.0" id="2">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <upgrade>
        <eid>6dbd99dc-f212-45de-a5f4-ddb78a2b5280</eid>
        <options>
          <force/>
        </options>
      </upgrade>
    </vzaenvm>
  </data>
</packet>
```

validate

Summary:

Validates a sample configuration.

Request specification:

Name	Min/Max	Type	Description
validate	1..1		
{			
config	1..1	env_configType	Sample configuration. Only the qos portion of the derived type venv_configType (p. 61) is actually used here.
}			

Returns:

Name	Min/Max	Type	Description
validation	1..1	validationType (p. 581)	Validation results. If the configuration is valid, the returned packet will be empty.

Description:

Sample configurations are used to create Virtuozzo Containers. A sample configuration contains QoS parameters that define how the system resources will be used. The values of the QoS parameters must satisfy the predefined formulae in order for the Container to function properly (you can find the formulae in the Agent dictionary). An individual formula usually has more than one QoS parameter, and an individual QoS parameter is usually referenced in more than one formula. So it is very important that the values of all the QoS parameters are set up correctly. The `validate` call will evaluate the current QoS parameter values using the existing formulae. If any of the formulae is violated, the return will contain the formula and the names of the invalid QoS parameters. You should always use this call to validate a new sample configuration before using it to create Virtuozzo Containers.

Example:

Submitting a Virtuozzo Container configuration for validation. Note the `dcachesize` QoS parameter. It contains a random value that we put there intentionally for the purpose of getting a negative validation result as a demonstration.

Input

```
<packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.0.0">
  <target>vzaenvm</target>
  <data>
    <vzaenvm>
      <validate>
        <config xsi:type="ns2:venv_configType">
          <qos>
            <id>avnumproc</id>
            <hard>96</hard>
          </qos>
          <qos>
            <id>cpuunits</id>
            <hard>6677</hard>
          </qos>
          <qos>
            <id>dcachesize</id>
            <hard>222222222222</hard>
            <soft>1717933</soft>
          </qos>
          <qos>
            <id>dgramrcvbuf</id>
            <hard>497780</hard>
            <soft>497780</soft>
          </qos>
          <qos>
            <id>diskinodes</id>
            <hard>88000</hard>
            <soft>80000</soft>
          </qos>
          <qos>
            <id>diskspace</id>
            <hard>225280</hard>
            <soft>204799</soft>
          </qos>
          <qos>
            <id>kmemsize</id>
            <hard>8692068</hard>
            <soft>7901880</soft>
          </qos>
```

```
          <qos>
            <id>lockedpages</id>
            <hard>385</hard>
            <soft>385</soft>
          </qos>
          <qos>
            <id>numfile</id>
            <hard>3072</hard>
          </qos>
          <qos>
            <id>numflock</id>
            <hard>337</hard>
            <soft>307</soft>
          </qos>
          <qos>
            <id>numiptent</id>
            <hard>100</hard>
          </qos>
          <qos>
            <id>numothersock</id>
            <hard>400</hard>
          </qos>
          <qos>
            <id>numproc</id>
            <hard>400</hard>
          </qos>
          <qos>
            <id>numpty</id>
            <hard>40</hard>
          </qos>
          <qos>
            <id>numsiginfo</id>
            <hard>1024</hard>
          </qos>
          <qos>
            <id>numtcpsock</id>
            <hard>400</hard>
          </qos>
          <qos>
            <id>oomguarpages</id>
            <hard>2147483647</hard>
            <soft>11701</soft>
          </qos>
          <qos>
            <id>othersockbuf</id>
            <hard>2136180</hard>
            <soft>497780</soft>
          </qos>
          <qos>
            <id>physpages</id>
            <hard>2147483647</hard>
            <soft>0</soft>
          </qos>
          <qos>
            <id>privvmpages</id>
            <hard>12871</hard>
            <soft>11701</soft>
          </qos>
          <qos>
            <id>shmpages</id>
            <hard>1170</hard>
          </qos>
          <qos>
            <id>tcprcvbuf</id>
            <hard>2633960</hard>
            <soft>995560</soft>
          </qos>
```

```
          <qos>
            <id>tcpsndbuf</id>
            <hard>2633960</hard>
            <soft>995560</soft>
          </qos>
          <qos>
            <id>vmguarpages</id>
            <hard>2147483647</hard>
            <soft>11701</soft>
          </qos>
        </config>
      </validate>
    </vzaenvm>
  </data>
</packet>
```

Output

The configuration that we submitted did not pass validation because the `dcachesize` parameter contained an incorrect value. The result set contains all of the affected parameters, the description of the problem, and the validation formula that was used.

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzaenvm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="24c462f8c2bt99re14"
time="2007-04-25T19:51:05+0000" priority="0" version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzaenvm>
    <ns2:validation>
      <ns2:type>0</ns2:type>
      <ns2:qos_id>avnumproc</ns2:qos_id>
      <ns2:qos_id>kmemsize</ns2:qos_id>
      <ns2:qos_id>dcachesize</ns2:qos_id>
      <ns2:warning>This constraint is important for reliable work of
applications in the Virtual Private Server. If it is not satisfied ,
applications will start to fail at the middle of operations instead of failing
at the moment of spawning more processes, and the application abilities to
handle resource shortage will be very limited.</ns2:warning>
      <ns2:formula>kmemsize.soft &amp;gt;= 40kb * avnumproc +
dcachesize.hard</ns2:formula>
    </ns2:validation>
  </ns2:vzaenvm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

If a configuration is valid, you should receive a packet similar to the following example.

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzaenvm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="23c462f8c22tf3ere14"
time="2007-04-25T19:50:58+0000" priority="0" version="4.0.0">
<ns1:origin>vzaenvm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
```

```
</ns1:dst>
<ns1:data>
  <ns2:vzaenvm/>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

vzarelocator

Purpose:

Virtuozzo Container migration interface.

Specification:

Derived from the `relocator` interface (p. 378)

Types

clone_optionsType

Summary:

The Container cloning options.

Type specification:

Extends `relocator:clone_optionsType (p. 379)`

Adds the following elements:

Name	Min/Max	Type	Description
`fast`	`0..1`	`none`	*This parameter is not currently used.*

`config`	`0..1`	`venv_configType` (p. 61)	Custom configuration parameters to apply to the resulting Containers.
			Currently, you can only set the names of the private area and root directories. If you are creating more than one clone, you can only use the default names as shown in the following example:
			`/vz/root/$VEID`
			`/vz/private/$VEID`
			To set other configuration parameters such as name, hostname, IP address, etc., use the `vzaenvm/set` call (p. 217) on each Container individually after the cloning operation is complete.

veid	0..1	veid_type (p. 568)	Container IDs to use for the clones. If you are creating more than one clone but the number of VEIDs specified here is less than the number of clones (i.e. not enough VEIDs for all of them), the VEIDs for some of the Containers will be allocated automatically from the VEID pool. If the element is omitted, every resulting Container will have an automatically assigned VEID.

hw_notesType

Summary:

Contains information about the files and directories that should be excluded from migration, and the list of warning messages. The structure is used in the `calc_env_config` call (p. 393) where it is returned together with the Container configuration information.

Type specification:

Extends `relocator:hw_notesType (p. 382)`

Adds the following elements:

Name	Min/Max	Type	Description
exclude	1..[]		A list of files and directories to be excluded from migration.
{			
path	1..1	base64Binary	Pathname.
discardable	1..1	boolean	Indicates whether the exclusion is optional or not: 0 -- indicates that it is critical to exclude the path from migration. 1 -- indicates that the exclusion is recommended but is optional. You will have to decide yourself whether it is really necessary to exclude the specified file or directory.
}			
warning	1..[]		A list of warning messages.
{			
message	1..1	string	Message text.
code	1..1	int	Code.
}			

Calls

Call	Description
migrate_p2v (p. 383)	Physical to virtual migration.
migrate_v2v (p. 389)	Virtual to virtual migration.
migrate_v2p (p. 392)	Virtual to physical migration.
calc_env_config (p. 393)	Calculates configuration parameters for for P2V migration.
move (p. 397)	Moves the environment's private area to a different location.
clone (p. 398)	Clones an environment.

vzanetworkm

Purpose:

The `vzanetworkm` interface allows to configure Virtuozzo virtual network settings and network traffic shaping on the Hardware Node. The virtual network devices that can be managed through this interface include virtual LAN adapters (or VLAN adapters) and network bridges. The network traffic shaping is managed by configuring Virtuozzo network classes (IP address ranges) and network interfaces to use these classes. The network settings inside Virtuozzo Virtual Environments are managed through the `envm` interface (p. 193).

Specification:

Derived from the `networkm` interface (p. 295).

Types

class_rateType

Summary:

Contains network traffic rates assigned to a Virtuozzo network class.

Type specification:

Name	Min/Max	Type	Description
`id`	`1..1`	`string`	Network class ID.
`rate`	`0..1`	`long`	Traffic rate value in Kbps.
`totalrate`	`0..1`	`long`	Total traffic rate value in Kbps.

net_configType

Summary:

Contains network traffic shaping configuration information.

Type specification:

Name	Min/Max	Type	Description
shaping	0..1	none	If present, traffic shaping is turned on.
range	0..[]		A list of Virtuozzo network classes. A class is a range of IP addresses for which you would like to manage traffic shaping.
{			
class_id	1..1	string	Class ID. An arbitrary string.
ip_address	1..1	ip_addressType (p. 42)	IP address range assigned to the class. The range is defined by specifying the IP address and a subnet mask.
comment	1..1	base64Binary	A free form comment.
}			
class	0..[]	class_rateType (p. 606)	Class-level network traffic rates.
interface	0..[]		Network interface-level bandwidth information.
{			
net_device_id	1..1	string	Network interface ID.
bandwidth	1..1	long	Bandwidth in Kbps.
}			

Calls

Call	Description
add (p. 296)	Adds a VLAN adapter or a network bridge to the Hardware Node.

list (p. 301)	Lists network devices installed on the Hardware Node, including physical and VLAN adapters, and networks bridges.
set (p. 307)	Modifies the network device settings.
del (p. 311)	Removes a network device.
set_config (p. 608)	Configures traffic shaping on a Virtuozzo network.
get_config (p. 610)	Retrieves traffic shaping information.

set_config

Summary:

Configures traffic shaping on a Virtuozzo network.

Request specification:

Name	Min/Max	Type	Description
`set_config`			
`{`			
`net_config`	`1..1`	`net_configType (p. 607)`	Network traffic shaping configuration information.
`}`			

Returns:

OK/Error.

Description:

The `set_config` call allows to perform the following tasks:

- Turning traffic shaping on or off.
- Creating Virtuozzo network classes. A network class is a range of IP addresses for which traffic shaping can be performed.
- Setting bandwidth limits for network interfaces.

Traffic shaping configuration data is stored in a file on the Hardware Node. When you call `set_config`, the old information contained in the file is deleted first and then the new information completely replaces it. This means that if you you don't want to loose the information contained in the file already, you must first retrieve it using the `get_config` cal (p. 610)l , update or add new entries to the retrieved structure, and then pass the structure to `set_config`.

For more information on Virtuozzo network shaping, please refer to Virtuozzo User's Guide.

Example 1:

Adding a network class and assigning the IP addresses in the range from `192.168.0.0` to `192.168.255.255` to it. At the same time, setting the traffic rate values for the new class and a bandwidth limit for the `etho` network interface.

```
<packet version="4.0.0" id="2">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <set_config>
        <net_config>
          <range>
            <class_id>1</class_id>
            <ip_address>
              <ip>192.168.0.0</ip>
              <netmask>255.255.0.0</netmask>
            </ip_address>
          </range>
          <class>
            <id>1</id>
            <rate>8</rate>
            <totalrate>1000</totalrate>
          </class>
          <interface>
            <net_device_id>etho</net_device_id>
            <bandwidth>102400</bandwidth>
          </interface>
        </net_config>
      </set_config>
    </vzanetworkm>
  </data>
</packet>
```

Example 2

Turning traffic shaping on.

```
<packet version="4.0.0" id="2">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <set_config>
        <net_config>
          <shaping/>
        </net_config>
      </set_config>
    </vzanetworkm>
  </data>
</packet>
```

Example 3

Turning traffic shaping off.

```
<packet version="4.0.0" id="2">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <set_config>
        <net_config/>
      </set_config>
    </vzanetworkm>
  </data>
</packet>
```

get_config

Summary:

Retrieves traffic shaping configuration information.

Request specification:

Name	Min/Max	Type	Description
get_config	1..1	none	

Returns:

Name	Min/Max	Type	Description
net_config	1..1	net_configType (p. 607)	Network traffic shaping configuration information.

Example:

Retrieving traffic shaping configuration information from the specified Container. The information is stored in a file on the Hardware Node. When Virtuozzo Containers is installed, the file contains some standard default values. Some of these values may not take full advantage of the resources provided by your system. For example, the standard default network interface bandwidth is 102400 Kbps. If you have a faster adapter in your system, you may want to change this value to reflect the actual adapter capabilities. Use the `set_config` call (p. 608) to modify the current traffic shaping configuration.

Input

```
<packet version="4.0.0">
  <target>vzanetworkm</target>
  <data>
    <vzanetworkm>
      <get_config/>
    </vzanetworkm>
  </data>
</packet>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?><ns1:packet
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzanetworkm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="17c46015272t99rf08"
time="2007-03-17T00:16:03+0000" priority="0" version="4.0.0">
<ns1:origin>vzanetworkm</ns1:origin>
<ns1:target>vzclient3</ns1:target>
<ns1:dst>
  <director>gend</director>
</ns1:dst>
<ns1:data>
  <ns2:vzanetworkm>
    <ns2:config>
      <ns2:range>
        <ns2:class_id>1</ns2:class_id>
        <ns2:ip_address>
          <ns2:netmask>0.0.0.0</ns2:netmask>
          <ns2:ip>0.0.0.0</ns2:ip>
        </ns2:ip_address>
        <ns2:comment></ns2:comment>
      </ns2:range>
      <ns2:interface>
        <ns2:net_device_id>eth0</ns2:net_device_id>
        <ns2:class>
          <ns2:id>1</ns2:id>
          <ns2:rate>8</ns2:rate>
          <ns2:totalrate>4096</ns2:totalrate>
        </ns2:class>
        <ns2:bandwidth>102400</ns2:bandwidth>
      </ns2:interface>
    </ns2:config>
  </ns2:vzanetworkm>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

vzapackagem

Purpose:

Virtuozzo template management interface. Provides a set of calls for Virtuozzo template installation and maintenance.

Specification:

Derived from the `packagem` interface (p. 318).

Calls

Call	Description
install (p. 323)	Installs a package or a template.
remove (p. 328)	Removes a package or a template.
update (p. 332)	Updates a package or a template.
list (p. 337)	Lists available templates and packages.
get_info (p. 343)	Obtain information about a package or template.
clean (p. 346)	Cleans the package manager cache.
fetch (p. 348)	Downloads EZ OS template packages from the remote repository to the local cache on the Hardware Node.
migrate (p. 351)	Migrates a template from one Hardware Node to another.

vzaproc_info

Purpose:

Process monitor.

Calls

Call	Description
start_monitor (p. 613)	Starts the monitor.
stop_monitor (p. 617)	Stop the monitor.
get (p. 617)	Retrieves a list of running processes.

start_monitor

Summary:

Starts the process monitor.

Request specification:

Name	Min/Max	Type	Description
`start_monitor`			
`{`			
`eid`	`1..1`	`eid_type` (p. 25)	Server ID.
`period`	`1..1`	`int`	Reporting period in seconds.
`key`	`0..1`	`string`	The parameter to order the result set by. For the list of parameters see Description below.
`limit`	`0..1`	`int`	Maximum number of processes to include in the report.
`descending`	`0..1`		If present, the list will be ordered in descending order.
`}`			

Returns:

Name	Min/Max	Type	Description
`ps_info`	`1..1`	`ps_infoType` (p. 53)	Processes information.

Description:

The call starts the process monitor in the specified server. The monitor sends the collected data back to the client at the specified time intervals until the client stops the monitor (p. 617) or disconnects from Agent. Only one process monitor can be running for a given connection.

The following table lists the parameters that can be specified in the `key` element to sort the result set by.

Parameter	Type	Description
`pid`	int	The process ID.
`user`	string	The user who has launched the process.
`pri`	int	The kernel scheduling priority for the process.
`ni`	int	The 'nice' parameter value defining the overall scheduling priority for the process. The less the 'nice' value, the higher the process priority.

rss	int	The total amount of physical memory used by the process, in kilobytes.
stat	string	The process current status. Can be 'R' (running), 'S' (sleeping, waiting for 'wake-up call'), 'D' (uninterruptable sleep), 'Z' (zombie, waiting for parent process), 'T' (stopped or traced). Sometimes the second symbol may appear: 'W' (process swapping), 'N' ('nice' process), 'L' (process has pages locked into memory). If the < sequence is displayed after the status, it means that this information was returned by the Parallels Agent software which, in turn, got this information from the 'ps' tool.
%cpu	float	The CPU time, in percent, used by the process.
%mem	float	The amount of physical memory, in megabytes, used by the process.
time	string	The total CPU time the process has used since its launch.
command	string	The command that invoked the process.

Example:

Input

Starting a process monitor on the specified server. Setting the reporting period at 10 seconds.

```
<packet version="4.0.0">
  <target>vzaproc_info</target>
  <data>
    <vzaproc_info>
      <start_monitor>
        <eid>9bafbeb7-85f7-499e-a210-40e00850a5f3</eid>
        <period>10</period>
      </start_monitor>
    </vzaproc_info>
  </data>
</packet>
```

Output

The very first message received from the monitor. It indicates that the monitor was started successfully.

```
<ns1:packet xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzaproc_info"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="49c45f69530t1a49rdfc" time="2007-03-11T06:33:23+0000" priority="0"
version="4.0.0">
<ns1:origin>vzaproc_info</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzaproc_info>
    <ns2:id>9bafbeb7-85f7-499e-a210-40e00850a5f3</ns2:id>
  </ns2:vzaproc_info>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

Output

One of the actual reports received from the monitor.

```
<ns1:packet xmlns:ns2="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="49c45f69530t1a49rdfc" time="2007-03-11T06:33:23+0000" priority="0"
version="4.0.0">
<ns1:origin>vzaproc_info</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns1:vzaproc_info>
    <ns1:ps_info xsi:type="ns2:ps_infoType">
      <ns2:process>
        <ns2:pid>1</ns2:pid>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>IGluaXQgWzNdICAgICAg</ns2:param>
```

```
        <ns2:param>MA==</ns2:param>
        <ns2:param>MjE=</ns2:param>
        <ns2:param>NDky</ns2:param>
        <ns2:param>U3M=</ns2:param>
        <ns2:param>MDA6MDA6MDg=</ns2:param>
        <ns2:param>MA==</ns2:param>
      </ns2:process>
      <ns2:process>
        <ns2:pid>28222</ns2:pid>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>IHN5c2xvZ2QgLW0gMA==</ns2:param>
        <ns2:param>MA==</ns2:param>
        <ns2:param>MjE=</ns2:param>
        <ns2:param>NTcy</ns2:param>
        <ns2:param>U3M=</ns2:param>
        <ns2:param>MDA6MDA6MDE=</ns2:param>
        <ns2:param>MA==</ns2:param>
      </ns2:process>
      <ns2:process>
        <ns2:pid>28247</ns2:pid>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>MC4x</ns2:param>
        <ns2:param>IC91c3Ivc2Jpbi9zc2hk</ns2:param>
        <ns2:param>MA==</ns2:param>
        <ns2:param>MTQ=</ns2:param>
        <ns2:param>MTAwNA==</ns2:param>
        <ns2:param>U3M=</ns2:param>
        <ns2:param>MDA6MDA6MDA=</ns2:param>
        <ns2:param>MA==</ns2:param>
      </ns2:process>
      <ns2:process>
        <ns2:pid>28257</ns2:pid>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>MC4w</ns2:param>

<ns2:param>IHhpbmV0ZCAtc3RheWFsaXZlIC1waWRmaWxlIC92YXIvcnVuL3hpbmV0ZC5waWQ=</n
s2:param>
        <ns2:param>MA==</ns2:param>
        <ns2:param>MTQ=</ns2:param>
        <ns2:param>NzUy</ns2:param>
        <ns2:param>U3M=</ns2:param>
        <ns2:param>MDA6MDA6MDA=</ns2:param>
        <ns2:param>MA==</ns2:param>
      </ns2:process>
      <ns2:process>
        <ns2:pid>28266</ns2:pid>
        <ns2:param>MC4w</ns2:param>
        <ns2:param>MC4x</ns2:param>
        <ns2:param>IGNyb25k</ns2:param>
        <ns2:param>MA==</ns2:param>
        <ns2:param>MTY=</ns2:param>
        <ns2:param>ODc2</ns2:param>
        <ns2:param>U3M=</ns2:param>
        <ns2:param>MDA6MDA6MDM=</ns2:param>
        <ns2:param>MA==</ns2:param>
      </ns2:process>
      <ns2:param_id>%cpu</ns2:param_id>
      <ns2:param_id>%mem</ns2:param_id>
      <ns2:param_id>command</ns2:param_id>
      <ns2:param_id>ni</ns2:param_id>
      <ns2:param_id>pri</ns2:param_id>
      <ns2:param_id>rss</ns2:param_id>
      <ns2:param_id>stat</ns2:param_id>
      <ns2:param_id>time</ns2:param_id>
      <ns2:param_id>user</ns2:param_id>
    </ns1:ps_info>
```

```
  </ns1:vzaproc_info>
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

stop_monitor

Summary:

Stops the process monitor.

Request specification:

Name	Min/Max	Type	Description
stop_monitor	1..1	none	

Description:

The call stops the process monitor that was started previuosly with the `start_monitor` call (p. 617).

Example:

```
<packet version="4.0.0">
  <target>vzaproc_info</target>
  <data>
    <vzaproc_info>
      <stop_monitor/>
    </vzaproc_info>
  </data>
</packet>
```

get

Summary:

Retrieves a list of processes running in the specified server.

Request specification:

Name	Min/Max	Type	Description
get	1..1		
{			
eid	1..1	eid_type (p. 25)	Server ID.
key	0..1	string	Parameter to order the result set by. For the list of parameters, see `start_monitor` (p. 613).
limit	0..1	int	Maximum number of processes to include in the list.
descending	0..1	none	If present, the resulting list will be sorted in descending order.

}			

Returns:

Name	Min/Max	Type	Description
ps_info	0..1	ps_infoType (p. 53)	The list of processes.

Example:

Input

```
<packet version="4.0.0">
  <target>vzaproc_info</target>
  <data>
    <vzaproc_info>
      <get>
        <eid>9bafbeb7-85f7-499e-a210-40e00850a5f3</eid>
        <limit>2</limit>
      </get>
    </vzaproc_info>
  </data>
</packet>
```

Output

```
<ns1:packet xmlns:ns3="http://www.swsoft.com/webservices/vzl/4.0.0/types"
xmlns:ns1="http://www.swsoft.com/webservices/vzl/4.0.0/protocol"
xmlns:ns2="http://www.swsoft.com/webservices/vza/4.0.0/vzaproc_info"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="4ec45f697ect5f49rdfc" time="2007-03-11T06:40:57+0000" priority="0"
version="4.0.0">
<ns1:origin>vzaproc_info</ns1:origin>
<ns1:target>vzclient6</ns1:target>
<ns1:dst>
  <ns1:director>gend</ns1:director>
</ns1:dst>
<ns1:data>
  <ns2:vzaproc_info>
    <ns2:ps_info xsi:type="ns3:ps_infoType">
      <ns3:process>
        <ns3:pid>1</ns3:pid>
        <ns3:param>MC4w</ns3:param>
        <ns3:param>MC4w</ns3:param>
        <ns3:param>IGluaXQgWzNdICAgICAg</ns3:param>
        <ns3:param>MA==</ns3:param>
        <ns3:param>MjM=</ns3:param>
        <ns3:param>NDky</ns3:param>
        <ns3:param>U3M=</ns3:param>
        <ns3:param>MDA6MDA6MDg=</ns3:param>
        <ns3:param>MA==</ns3:param>
      </ns3:process>
      <ns3:process>
        <ns3:pid>28222</ns3:pid>
        <ns3:param>MC4w</ns3:param>
        <ns3:param>MC4w</ns3:param>
        <ns3:param>IHN5c2xvZ2QgLW0gMA==</ns3:param>
        <ns3:param>MA==</ns3:param>
        <ns3:param>MjE=</ns3:param>
        <ns3:param>NTcy</ns3:param>
        <ns3:param>U3M=</ns3:param>
        <ns3:param>MDA6MDA6MDE=</ns3:param>
        <ns3:param>MA==</ns3:param>
      </ns3:process>
      <ns3:param_id>%cpu</ns3:param_id>
      <ns3:param_id>%mem</ns3:param_id>
      <ns3:param_id>command</ns3:param_id>
      <ns3:param_id>ni</ns3:param_id>
      <ns3:param_id>pri</ns3:param_id>
      <ns3:param_id>rss</ns3:param_id>
      <ns3:param_id>stat</ns3:param_id>
      <ns3:param_id>time</ns3:param_id>
      <ns3:param_id>user</ns3:param_id>
    </ns2:ps_info>
  </ns2:vzaproc_info>
```

```
</ns1:data>
<ns1:src>
  <ns1:director>gend</ns1:director>
</ns1:src>
</ns1:packet>
```

vzaprocessm

Purpose:

System processes management.

Calls

Call	Description
kill (p. 620)	Send a signal to the specified process.

kill

Summary:

Sends a signal to the specified process in the specified Virtuozzo Container.

Request specification:

Name	Min/Max	Type	Description
`kill`			
`{`			
`  eid`	`1..1`	`eid_type (p. 25)`	Server ID.
`  pid`	`1..[]`	`int`	Process ID.
`  signal`	`1..1`	`int`	Signal number.
`}`			

Returns:

OK/Error

Description:

The signal number can be one from following:

```
1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL
5) SIGTRAP 6) SIGABRT 7) SIGBUS 8) SIGFPE
9) SIGKILL 10) SIGUSR1 11) SIGSEGV 12) SIGUSR2
13) SIGPIPE 14) SIGALRM 15) SIGTERM 17) SIGCHLD
18) SIGCONT 19) SIGSTOP 20) SIGTSTP 21) SIGTTIN
22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ
26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO
30) SIGPWR 31) SIGSYS 32) SIGRTMIN 33) SIGRTMIN+1
34) SIGRTMIN+2 35) SIGRTMIN+3 36) SIGRTMIN+4 37) SIGRTMIN+5
38) SIGRTMIN+6 39) SIGRTMIN+7 40) SIGRTMIN+8 41) SIGRTMIN+9
42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13
46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMAX-15 49) SIGRTMAX-14
50) SIGRTMAX-13 51) SIGRTMAX-12 52) SIGRTMAX-11 53) SIGRTMAX-10
54) SIGRTMAX-9 55) SIGRTMAX-8 56) SIGRTMAX-7 57) SIGRTMAX-6
58) SIGRTMAX-5 59) SIGRTMAX-4 60) SIGRTMAX-3 61) SIGRTMAX-2
62) SIGRTMAX-1 63) SIGRTMAX
```

Example:

Sending the SIGINT signal to process 7368 running in the Container specified by its Server ID.

```
<packet version="4.0.0">
  <target>proc</target>
  <data>
    <proc>
      <kill>
        <eid>37ea79c2-3565-0e43-887f-91d4678c843e</eid>
        <pid>7368</pid>
        <signal>2</signal>
      </kill>
    </proc>
  </data>
</packet>
```

vzaup2date

Purpose:

Updates Virtuozzo Containers software on the Hardware Node.

Types

configurationType

Summary:

Configuration of the Virtuozzo Containers update utility.

Type specification:

Name	Min/Max	Type	Description

{			
connection	0..1		
{			
server	1..1	string	Update server.
user	0..1	string	User name.
password	0..1	string	Password.
proxy	0..1		Proxy settings.
{			
server	1..1	string	Proxy server
user	0..1	string	User name.
password	0..1	string	Password.
}			
local_path	0..1	string	Path on the local machine to download the updates to.
log_path	0..1	string	Log file path.
}			
product			Product definition.
{			
id	1..1	string	Product ID.
name	1..1	base64Binary	Product name.
install	1..1		Type of installation.
[			*Denotes a* `choice` *element.*
manual	1..1	none	Install the updates manually.
automatic	1..1	none	Install the updates automatically.
disabled	1..1	none	Automatic update is disabled.
auto_download	1..1	none	Automatically download but don't install the updates.
]			
source	0..1		Location of the product update repository.
{			
value	1..1	base64Binary	Source URL. Path to XML file.
standard	0..1	base64Binary	Standard location.
}			

destination	0..1	base64Binary	Download location.
}			
service	0..1		Service settings.
{			
autoreboot	0..1	none	Enable automatic reboot.
check_period	0..1		Specified the day of the week on which to check for updates. If not specified -- check daily.
[			*Denotes a choice element.*
sunday	1..1	none	Sunday.
monday	1..1	none	Monday.
tuesday	1..1	none	Tuesday.
wednesday	1..1	none	Wednesday.
thursday	1..1	none	Thursday.
friday	1..1	none	Friday.
saturday	1..1	none	Saturday.
]			
hour	1..1	int	Specifies the hour on which to check for updates.
disabled	0..1	none	If present, indicates that the vzupsvc service is disabled.
}			
}			

updateType

Summary:

Virtuozzo Containers update parameter structure.

Type specification:

Name	Min/Max	Type	Description
id	0..1	string	Update ID.
name	0..1	base64Binary	Update name.
description	0..1	base64Binary	Description of the update.
size	0..1	long	Update size in bytes.

reboot	0..1	none	Indicates whether system reboot is needed after the update is installed.
version	0..1	string	Version of the update. If version is unknown, the value is "unknown". If no version is specified -- no updates are available for given item.
installed_version	0..1	string	Version of previous installed update for given item. If not specified -- the item is not installed.
date	0..1	datetime_type	Update date.

Calls

Call	Description
get_config (p. 624)	Returns current configuration of the Virtuozzo Containers update service.
set_config (p. 625)	Allows to modify the Virtuozzo Containers update service configuration.
list (p. 625)	Requests the list of the available Virtuozzo Containers updates.
install (p. 626)	Install Virtuozzo Containers updates.
uninstall (p. 626)	Allows to uninstall a Virtuozzo Containers update.

get_config

Summary:

Returns current configuration of the Virtuozzo Containers update service.

Request specification:

Name	Min/Max	Type	Description
get_config	1..1		

Returns:

Name	Min/Max	Type	Description
config	1..1	configurationType (p. 621)	Configuration info.

set_config

Summary:

Allows to modify the Virtuozzo Containers update service configuration.

Request specification:

Name	Min/Max	Type	Description
set_config			
{			
config	1..1	configurationType (p. 621)	Configuration information.
}			

Returns:

OK/Errors.

list

Summary:

Requests the list of the available Virtuozzo Containers updates.

Request specification:

Name	Min/Max	Type	Description
list			
{			
update	0..[]	updateType (p. 623)	Update information. If specified, gets the list of products according to the specified parameters. If the parameter is not specified, gets the list of all available updates.
}			

Returns:

Name	Min/Max	Type	Description
updates		updateType (p. 623)	A list of available updates.

install

Summary:

Install Virtuozzo Containers updates.

Request specification:

Name	Min/Max	Type	Description
`install`	`1..1`		
`{`			
`update`	`0..[]`		List of update identifiers for install. If the list is empty, installs all available updates.
`{`			
`id`	`0..1`	`string`	Update ID. To get the list of the available updates, use the `list` call (p. 625).
`name`	`0..1`	`base64Binary`	Update name.
`}`			
`no_reboot`	`0..1`	`none`	Include this element to suppress automatic rebooting of the system if it is required by the update being installed.
`no_boot_loader`	`0..1`	`string`	Do not automatically configure bootloader for a newly installed kernel (if any).
`}`			

Returns:

OK/Errors.

uninstall

Summary:

Allows to uninstall a Virtuozzo Containers update.

Request specification:

Name	Min/Max	Type	Description
`uninstall`	`1..1`		
`{`			
`update`	`1..[]`		List of update identifiers to uninstall.

{			
id	0..1	string	Update ID.
name	0..1	base64Binary	Update name.
}			
no_reboot	0..1	none	Include this element to suppress automatic rebooting of the system.
}			

Returns:

OK/Errors.

vzasupport

Purpose:

The Virtuozzo Support Tunnel interface. Support Tunnel is a Virtuozzo troubleshooting utility that can be used to diagnose and solve possible problems with your Virtuozzo Containers installation. The utility allows you to establish a private secure connection with SWsoft support server. The connection can be used by SWsoft support team to gain access to your Hardware Node in order to accurately diagnose and repair the problem. The communication is performed using a VPN (Virtual Private Network) tunnel. For more information on Virtuozzo Support Tunnel, please see Virtuozzo User's Guide.

All of the calls in this interface except the `problem_report` call (p. 632) are available on Virtuozzo for Linux only. The `problem_report` call (p. 632) is available on both Linux and Windows. For the information on how to use Virtuozzo Support Tunnel functionality on Windows, see Virtuozzo for Windows User's Guide.

Calls

Call	Description
start_channel (p. 628)	Establishes a VPN connection with SWsoft support server.
stop_channel (p. 629)	Closes a connection that was previously opened by the `start_channel` call.
get_channel_status (p. 630)	Obtains the status of the Virtuozzo Support Tunnel service.
set_key (p. 631)	Installs Virtuozzo Support Tunnel certificate key.
get_key_status (p. 631)	Obtains the certificate key status information.
remove_key (p. 632)	Removes Virtuozzo Support Tunnel certificate key.
problem_report (p. 632)	Allows to manually submit a problem report to SWsoft technical support.

start_channel

Available on Virtuozzo for Linux only.

Summary:

Establishes a VPN connection with SWsoft support server.

Request specification:

Name	Min/Max	Type	Description
`start_channel`			

Returns:

OK/Error

Description:

The following list describes the tasks that must be performed prior to establishing a VPN connection with SWsoft support server for the first time:

- Make sure the `openvpn` (version 2.0 and above) and `vzvpn` packages are installed on your Hardware Node. These packages are automatically installed during the installation of Virtuozzo Containers version 2.6.2 to 4.0. If you are running a version of Virtuozzo Containers older than 2.6.2, you may need to manually copy these packages and install them on your Hardware Node.
- Make sure that port 80 is opened on the Hardware Node.
- Edit the `/etc/vzvpn/vzvpn.conf` file to specify the correct parameters for your proxy server, if you use one. Detailed information on these parameters is given in the vzvpn Configuration File subsection of the Virtuozzo Reference Guide.
- Obtain a special certificate from SWsoft which will uniquely identify you as a Virtuozzo Containers user. Certificates are issued by SWsoft as files containing the certificate key. The key should be installed on your Node using the `set_key` call (p. 631). See Virtuozzo User's Guide for information on how to obtain a certificate from SWsoft.

After establishing a VPN connection, contact the SWsoft support team via telephone or e-mail and describe the problem with your Virtuozzo Containers installation. The support team will use the VPN connection that you've established to analyze and solve the problem.

When the connection is no longer needed, close it using the `stop_channel` call (p. 629).

Note: Virtuozzo Support Tunnel is implemented as a standard Linux service running in the background on the Hardware Node. To have this service running after your Hardware Node reboot, you should set it to the `autoboot` mode or start it manually by executing the `/etc/init.d/vzvpn start` command. To check the status of the Support Channel service, use the `get_channel_status` call (p. 630).

---

stop_channel

Available on Virtuozzo for Linux only.

Summary:

Closes a VPN connection that was previously opened by the `start_channel` call (p. 628).

Request specification:

Name	Min/Max	Type	Description
`stop_channel`			

Returns:

OK/Error.

get_channel_status

Available on Virtuozzo for Linux only.

Summary:

Obtains the status of the Virtuozzo Support Tunnel service.

Request specification:

Name	Min/Max	Type	Description
`get_channel_status`			

Returns:

Name	Min/Max	Type	Description
`status`	`1..1`	`boolean`	Virtuozzo Support Tunnel status information: `0` -- Support Tunnel is operational. This means that a VPN connection with SWsoft support server has been established. `1` -- The Support Tunnel service is running but no active VPN connection with SWsoft support server has been detected. `2` -- The Support Tunnel service is not running. You must start the service before using the Support Tunnel functionality. To start the service, execute the following command on the Hardware Node: `/etc/init.d/vzvpn start`

set_key

Available on Virtuozzo for Linux only.

Summary:

Installs Virtuozzo Support Tunnel certificate key.

Request specification:

Name	Min/Max	Type	Description
`support_channel`			
`{`			
`key`	`0..1`	`base64Binary`	Certificate key data. You obtain a certificate from SWsoft as a file containing the certificate key information. Read the data from the file in your client program and use it as input here.
`}`			

Returns:

OK/Error.

Description:

Before you can use Virtuozzo Support Tunnel functionality, you have to obtain a certificate from SWsoft. Support Tunnel certificates are provided as files containing a certificate key which uniquely identifies you as a Virtuozzo user on the SWsoft technical support server. For more information on how to obtain a certificate please see Virtuozzo User's Guide.

get_key_status

Available on Virtuozzo for Linux only.

Summary:

Reports the Virtuozzo Support Tunnel certificate status information.

Request specification:

Name	Min/Max	Type	Description
`get_key_status`			

Returns:

Name	Min/Max	Type	Description
key_status	1..1	int	Certificates/key status: 0 -- certificate key is installed. 1 -- certificates key is not installed 2 -- invalid certificate key.

Description:

Use this call to see if you have a valid certificate key installed on your Hardware Node. If the key is not installed or is invalid, you will have to obtain a new certificate from SWsoft before you can use the Virtuozzo Support Channel functionality.

remove_key

Available on Virtuozzo for Linux only.

Summary:

Removes Virtuozzo Support Tunnel certificate key from the Hardware Node.

Request specification:

Name	Min/Max	Type	Description
remove_key			

Returns:

OK/Error.

problem_report

Summary:

Allows to manually submit a problem report to SWsoft technical support. Accepts a problem description from a user, generates a system report and sends it directly to SWsoft technical support.

Request specification:

Name	Min/Max	Type	Description
problem_report			
{			
name	1..1	base64Binary	Report name (you can choose any name you like).
company	1..1	base64Binary	Your company name.
email	1..1	base64Binary	Your e-mail address.
subject	1..1	base64Binary	Subject.

problem_description	1..1	base64Binary	Problem description in your own words.
ticket	0..1	base64Binary	Ticket ID. If this is a new report, omit the element from the request. The new ticket ID will be generated by SWsoft support server. If the report is a follow up to a previous report, include the element containing the ticket ID of the original report.
}			

Returns:

Name	Min/Max	Type	Description
ticket	1..1	base64Binary	Ticket ID generated by SWsoft support server for the new report.

Description:

If there's a problem with your Virtuozzo system, you can use this call to send a troubleshooting request directly to SWsoft technical support. Enter your name, your company name, your e-mail address, the subject of the request, and the description of the problem. The problem_report call will collect all the necessary information about your Virtuozzo system including log data, Container settings, licence information, etc. It will then send the complete report to the SWsoft technical support server. The server will process the request and will send back a ticket ID confirming that the request has been processed and registered. Keep this ID for future reference. If later you decide to send an additional information regarding the same problem, supply the ticket ID in the subsequent request. The report will be reviewed by SWsoft technical support team and a solution will be sent to you as soon as possible.

CHAPTER 5

Parallels Server Types and Interfaces

This chapter describes the types and interfaces that are specific to the Parallels Server management only. The majority of the types and interfaces described here are derived by extension from the base types and interfaces (p. 23).

In This Chapter

Common Types.....634
Interfaces 642

Common Types

This subsection describes the common data types that are specific for Parallels Server.

Complex Types

Complex Types are custom data types that can contain data, attributes and other elements.

deviceType

Summary:

Contains a device settings.

Type specification:

Name	Min/Max	Type	Description
`enabled`	`1..1`	`boolean`	Determines if the specified device is enabled.
`connected`	`0..1`	`boolean`	Determines if the specified device is connected.
`emulation_type`	`1..1`	`int`	A virtual device emulation type.
`sys_name`	`0..1`	`string`	A virtual device system name. The value must be a UTF\-8 encoded, null\-terminated string.
`remote`	`0..1`	`boolean`	Determines if a device is remote (accessed from a virtual machine remotely) or not.

friendly_name	0..1	string	A virtual device user-friendly name. A UTF\-8 encoded, null\-terminated string).
summary_info	1..1	string	A device description.
is_bootable	0..1		Determines if a device is a boot device or not. Bootable devices are used for a virtual machine boot.
is_boot_in_use	0..1		Determines if a boot device is enabled or not. A boot device can be either enabled or disabled. If a device is disabled, it is ignored during the boot operation. If a boot device is enabled, a virtual machine will try to boot from the device according to a device sequence index.
boot_sequence_index	0..1	int	An index assigned to a boot device in the boot device priority list. When a virtual machine is powered on, it will first try to boot from the boot device that has a sequence index of 0 (zero). If the device is not bootable, it will try to boot from the device with index 1 (one), and so on forth.Device index is a property that, together with device type, is used to uniquely identify a device on a virtual machine.

Device types:

- PHT_VIRTUAL_DEV_DISPLAY
- PHT_VIRTUAL_DEV_KEYBOARD
- PHT_VIRTUAL_DEV_MOUSE
- PHT_VIRTUAL_DEV_FLOPPY
- PHT_VIRTUAL_DEV_HARD_DISK
- PHT_VIRTUAL_DEV_NET_ADAPTER
- PHT_VIRTUAL_DEV_PARALLEL_PORT
- PHT_VIRTUAL_DEV_SERIAL_PORT
- PHT_VIRTUAL_DEV_OPTICAL_DISK
- PHT_VIRTUAL_DEV_USB_DEVICE
- PHT_VIRTUAL_DEV_SOUND

guest_os_memory

Summary:

Guest OS virtual machine memory size requirements (optimal and recommended).

Type specification:

Name	Min/Max	Type	Description
`guest_os_platform`	`1..1`	`string`	Guest OS platform name.
`guest_os_name`	`1..1`	`string`	The name guest OS installed in a virtual machine.
`optimal_vm_memory_size`	`1..1`	`int`	The memory size optimal for a particular guest OS.
`min_recommended_vm_memory_size`	`1..1`	`int`	The minimum memory size needed for an individual virtual machine operation (in megabytes). By default, the same small value is currently used for all virtual machines regardless of the guest OS type.
`max_recommended_vm_memory_size`	`1..1`	`int`	The maximum memory size recommended for a virtual machine (in megabytes). The value must not exceed the reserved memory size.

venv_configType

Summary:

Contains virtual machine configuration information.

Type specification:

Extends `venv_configType` (p. 61)

Adds the following elements:

Name	Min/Max	Type	Description
`memory_size`	`0..1`	`int`	The RAM size to set, in megabytes.
`video_memory_size`	`0..1`	`int`	The video memory to set, in megabytes

cpu_count	0..1	int	The number of CPUs in the virtual machine.
cpu_mode	0..1	int	A virtual machine CPU mode (32 bit or 64 bit): ▪ zero (0) for 32 bit mode, ▪ 1 for 64 bit mode.
cpu_accel_level	0..1	int	A virtual machine CPU acceleration level: ▪ zero (0) - acceleration disabled, ▪ 1 - acceleration normal, ▪ 2 - acceleration high.
auto_start	0..1	int	Determines if the specified virtual machine is set to start automatically on Parallels server startup: ▪ zero(0) - autostart disabled, a virtual machine is started manually, ▪ 1 - a virtual machine starts automatically on Parallels server startup, ▪ 2 - a virtual machine starts automatically on Parallels server reload.
auto_start_delay	0..1	int	Sets the time delay that will be used during the virtual machine automatic startup, in seconds.
start_login_mode	0..1	int	Determines if the automatic startup login mode is set for a virtual machine: ▪ zero (0) start account, ▪ 1 - root account, ▪ 2 - user account.
start_user_login	0..1	string	User name used as login during a virtual machine automatic startup. Login must be in UTF-8 encoding.
start_user_password	0..1	string	Password used during a virtual machine automatic startup. Password must be in UTF-8 encoding.
auto_stop	0..1	int	Determines if the specified virtual machine is set to stop automatically on Parallels server stop.
window_mode	0..1	int	Determines a virtual machine window mode.

last_modified_date	0..1	datetime_type	Returns the date and time when the specified virtual machine was last modified. Returned time is a local time (i.e. the time value is already converted to local time zone).
last_modifier_name	0..1	string	Returns the name of the user who last modified the specified virtual machine.
guest_sharing_enabled	0..1	boolean	Determines if guest sharing is enabled (the guest OS disk drives are visible in the host OS).
guest_sharing_auto_mount	0..1	boolean	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. Parallels Tools must be installed to use this feature.
host_sharing_enabled	0..1	boolean	Determines if host sharing is enabled (host shared folders are visible in the guest OS). Parallels Tools must be installed to use this feature.
host_sharing_local	0..1	boolean	A virtual machine host OS local sharing enabling sign.
host_sharing_global	0..1	boolean	A virtual machine host OS global sharing enabling sign.
disk_cache_write_back	0..1	boolean	Determines if disk cache write-back is enabled for the specified virtual machine.
close_app_on_shutdown	0..1	boolean	Virtual Machine close app on shutdown sign.
system_flags	0..1	string	Sets Virtual Machine system flags. A specified string value must be in UTF-8 encoding and end with '0' terminating symbol.
foreground_priority	0..1	int	Sets a virtual machine foreground processes priority.
background_priority	0..1	int	Virtual machine background process priority type.
server_host_name	0..1	string	Hostname of the server hosting the specified virtual machine.

home_path	0..1	string	A virtual machine home directory name and path
icon	0..1	string	The name of the icon file used by the specified virtual machine.
show_task_bar	0..1	boolean	Determines if Windows task bar is displayed when the virtual machine runs in coherence mode. Parallels Tools must be installed to use this feature.
relocate_task_bar	0..1	boolean	Allows enabling or disabling the Windows task bar relocation feature. Parallels Tools must be installed to use this feature.
exclude_dock	0..1	boolean	Sets the 'exclude dock' option. When a virtual machine is running in coherence mode, you can make the dock stay always below the virtual machine window. If the window is moved below the dock, the portion of it that crossed that boundary will be cut (will not be redrawn). If the parameter is set to FALSE, the virtual machine window will be able to move below the dock. Parallels Tools must be installed to use this feature.
multi_display	0..1	boolean	Sets the virtual machine multi-display option.
additional_screen_resolutions_enabled	0..1	boolean	Determines if additional screen resolution support is enabled in a virtual machine.
os_screen_resolution_in_full_screen_mode_enabled	0..1	boolean	Determines if a virtual machine OS resolution is in full screen mode.
app_in_dock_mode	0..1	int	Determines the current dock mode for the specified virtual machine. Parallels Tools must be installed to use this feature.
dock_icon_type	0..1	int	A virtual machine dock icon type. This functionality is part of the Parallels Tools package.
vnc_mode	0..1	int	Sets the virtual machine VNC mode.
vnc_port	0..1	int	Sets or returns the VNC port number for the specified virtual machine.

vnc_nic	0..1	string	Sets the virtual machine VNC hostname.
vnc_password	0..1	string	Sets the virtual machine VNC password, in UTF-8 encoding.
disabled	0..1	boolean	
origin_sample	0..1	string	
tools_status	0..1	int	
access_for_others	0..1	int	
device_list	0..1		
{			
device	0..[]	deviceType (p. 634)	
}			

vt_settingsType

Summary:

Contains global Virtual Machine settings.

Type specification:

Extends vt_settingsType (p. 63)

Adds the following elements:

Name	Min/Max	Type	Description
default_vm_folder	0..1	string	Name and path of the directory in which new virtual machines are created by default.
default_user_vm_folder	0..1	string	Name and path of the default virtual machine directory for the specified user.
use_management_console	0..1	boolean	Determines if the user is allowed to use the Parallels Service management console utility.
can_change_server_settings	0..1	boolean	Determines if new users have the right to modify Parallels Service preferences.
auto_adjust_reserved_vm_memory_size	0..1	boolean	Determines whether memory allocation for Parallels Service is performed automatically or manually.

reserved_vm_memory_size	0..1	int	Determines the amount of physical memory reserved for Parallels Service operation.
min_vm_memory_size	0..1	int	Determines the minimum required memory size that must be allocated to an individual virtual machine.
max_vm_memory_size	0..1	int	Determines the maximum memory size that can be allocated to an individual virtual machine.
vm_memory	0..1	guest_os_memory (p. 636)	Guest OS virtual machine memory size requirements (optimal and recommended).
min_reserved_memory_limit	0..1	int	Determines the minimum amount of physical memory that must be reserved for Parallels Service operation.
max_reserved_memory_limit	0..1	int	Determines the maximum amount of physical memory that can be reserved for Parallels Service operation.
vnc_port	0..1	int	A virtual machine VNC port number.
vnc_nic	0..1	int	

Interfaces

The material in this section describes Parallels Server API interfaces. The term *interface*, as we use it, is somewhat similar to a class in object-oriented programming. We use interfaces to group related data types (structures) and calls (methods). The data types and calls are defined using XML Schema language (XSD). The body of an Agent XML request always begins with the name of an interface followed by the name of a call. The rest of the request body is composed according to the call specifications.

The interfaces described in this chapter provide Parallels Server management functionality. Please note that the interfaces described here do not comprise a complete list of Parallels Server functions. For the complete list please also see the Base Types and Interfaces chapter (p. 23).

vzpenvm

Purpose:

Virtual machine management interface.

Specification:

Derived from the `envm` interface (p. 193)

Calls

install_tools

Summary:

Installs Parallels tools for a specified virtual machine.

Request specification:

Name	Min/Max	Type	Description
`install_tools`			
`{`			
`  eid`	`1..1`	`eid_type (p. 25)`	Server ID
`}`			

Returns:

OK/Error

pause

Summary:

Pauses a virtual machine.

Request specification:

Name	Min/Max	Type	Description
pause			
{			
eid	1..1	eid_type (p. 25)	Server ID
}			

Returns:

OK/Error

reg

Summary:

Registers an existing virtual machine in Parallels server virtual machine directory.

Request specification:

Name	Min/Max	Type	Description
reg			
{			
path	1..1	string	Path to existing virtual machine configuration file.
}			

Returns:

Returns new virtual machine configuration.

unreg

Summary:

Unregisters an existing virtual machine from Parallels server.

Request specification:

Name	Min/Max	Type	Description
unreg			

{			
eid	1..1	eid_type (p. 25)	Server ID
}			

Returns:

OK/Error

update_device

Summary:

Set device properties on a running virtual machine.

Request specification:

Name	Min/Max	Type	Description
update_device			
{			
eid	1..1	eid_type (p. 25)	Server ID of virtual machine a device is to be updated for.
sys_name	1..1	string	A virtual device system name. The value must be a UTF\-8 encoded, null\-terminated string.
device	1..1	deviceType (p. 634)	Device configuration with updated properties.
}			

Returns:

OK/Error

vzprelocator

Types

clone_optionsType

Summary:

Additional options for a virtual machine cloning operation executed using the clone (p. 646) call.

Type specification:

Name	Min/Max	Type	Description
`name`	`1..1`	`string`	Cloned virtual machine name. If only one clone is created, then this name is assigned to a clone. If the number of clones is greater than 1, then clones are assigned name1, name2, and so on.
`config`	`0..1`	`venv_configType (p. 636)`	A custom configuration used to apply customization to a cloned virtual machine(s) configuration.
`force`	`0..1`	`boolean`	If specified, then running virtual machine is forcedly stopped before cloning begins and restarted after cloning is finished.

Calls

clone

Summary:

Creates one or more clones of an existing virtual machine.

Request specification:

Name	Min/Max	Type	Description
clone			
{			
eid	1..1	eid_type (p. 25)	Server ID
count	1..1	int	The number of a virtual machine clones to create.
options	1..1	clone_optionsType	Additional options passed to the clone call.
}			

Returns:

eid_list

Name	Min/Max	Type	Description
eid_list	1..1	eid_listType (p. 32)	A list containing the Server IDs of the created virtual machines.

CHAPTER 6

Appendix A: Performance Counters

Performance Classes

There are two groups of performance classes: one is for monitoring Virtuozzo Containers, and the other is for monitoring the host server (Hardware Node). Both groups are listed in the following table.

Class ID	Resource Type	Class Instances
Container classes		
counters_vz_cpu	CPU	N/A
counters_vz_ubc	UBC (User Bean Counters).	N/A
counters_vz_net	Container network.	Use vznetstat command-line utility to obtain a list of instances. The Net.Class column will contain the available instances. The rows with CTID = 0 (Container 0 or Hardware Node) are not applicable.
counters_vz_quota	Disk quota.	N/A
counters_vz_loadavg	Load average.	N/A
counters_vz_system	System info.	N/A
counters_vz_slm	SLM	N/A
counters_vz_memory	Memory	N/A
counters_vz_hw_net	Hardware Node network.	Use vznetstat command-line utility to obtain a list of instances. The Net.Class column will contain the available instances. Only the rows with CTID=0 (Container 0 or Hardware Node) must be looked at.
Hardware Node classes		
counters_cpu	CPU	N/A
counters_disk	Disk	The name of the hard disk device.
counters_memory	Memory	N/A
counters_net	Network	The name of the network interface.

counters_loadavg	CPU	N/A
counters_system	System info.	N/A

Performance Counters

Note: UBC failcounters are not supported in the current version of Parallels Agent.

The tables below contain lists of performance counters by their parent class. The table columns are:

Counter ID — Counter ID. The IDs are used in Agent calls as input/output parameters.

Value — The data type of the counter value(s).

Type — Counter type. A performance counter may be one of the following types:

- *Periodic counter (type 0)*. Contains the minimum, maximum, and average values for the given time period.
- *Incremental counter (type 1)*. The value of an incremental counter is always higher or equals to the previous value. A good example is a network counter that counts the number of bytes the interface has sent or received. The minimum, maximum, and average values are the same and represent the difference between the current value and the value from the previous report.
- *Cumulative counter (type 2)*. The minimum, maximum, and average values are the same and represent the total accumulated value since the server was started. On server restart, counter values are reset to zero.

Units — Units of measure (bytes, percent, seconds, pieces, etc.)

Description — Counter description.

CPU counters (counters_vz_cpu class)

Counter ID	Value	Type	Units	Description
counter_cpu_system	int	2	seconds	System CPU time.
counter_cpu_user	int	2	seconds	User CPU time.
counter_cpu_idle	int	2	seconds	Idle CPU time.
counter_cpu_nice	int	2	seconds	Nice CPU time.
counter_cpu_starvation	int	2	seconds	'Starvation' CPU time (i.e. the difference between the guaranteed and used CPU time).
counter_cpu_system_states	int	0	percent	System CPU time in percent.
counter_cpu_user_states	int	0	percent	User CPU time in percent.

counter_cpu_idle_states	int	0	percent	Idle CPU time in percent.
counter_cpu_nice_states	int	0	percent	Nice CPU time in percent.
counter_cpu_starvation_states	int	0	percent	Starvation CPU time in percent.
counter_cpu_used	float	0	percent	Total CPU usage in percent.
counter_cpu_share_used	float	0	percent	The real CPU usage of the Container against the CPU limit set for this Container.
counter_cpu_limit	float	0	percent	The share of the CPU time the Container may never exceed.

UBC counters (counters_vz_ubc class)

Counter ID	Value	Type	Units	Description
numproc	int	0	pcs	Number of processes and kernel-level threads.
numtcpsock	int	0	pcs	Number of TCP sockets.
numothersock	int	0	pcs	Number of non-TCP sockets.
vmguarpages	int	0	4K-pages	Memory allocation guarantee.
kmemsize	int	0	bytes	Size of non-swappable kernel memory.
tcpsndbuf	int	0	bytes	Total size of 'send' buffers for TCP sockets.
tcprcvbuf	int	0	bytes	Total size of 'receive' buffers for TCP sockets
othersockbuf	int	0	bytes	Total size of UNIX-domain socket buffers, UDP, and other datagram protocols 'send' buffers.
dgramrcvbuf	int	0	bytes	Total size of 'receive' buffers of UDP and other datagram protocols.
oomguarpages	int	0	4K-pages	Out-of-memory guarantee.
privvmpages	int	0	4K-pages	Size of the Container private memory.

lockedpages	int	0	4K-pages	Memory not allowed to be swapped out.
shmpages	int	0	4K-pages	Size of the shared memory.
physpages	int	0	4K-pages	Total size of RAM used by Container processes.
numfile	int	0	pcs	Number of open files.
numflock	int	0	pcs	Number of file locks.
numpty	int	0	pcs	Number of pseudo-terminals.
numsiginfo	int	0	pcs	Number of 'siginfo' structures.
dcachesize	int	0	bytes	Total size of 'dentry' and 'inode' structures locked in memory.
numiptent	int	0	pcs	Number of IP packet filtering entries.

Network counters (counters_vz_net class)

Counter ID	Value	Type	Units	Description
counter_net_incoming_bytes	int	2	bytes	The amount of incoming network traffic in bytes.
counter_net_incoming_packets	int	2	pcs	The amount of incoming network traffic in packets.
counter_net_outgoing_bytes	int	2	bytes	The amount of outgoing network traffic in bytes.
counter_net_outgoing_packets	int	2	pcs	The amount of outgoing network traffic in packets.

Disk quota counters (counters_vz_quota class)

Counter ID	Value	Type	Units	Description
diskspace	int	0	1K-blocks	The total size of disk consumed by the Container.
diskspace_hard	int	0	1K-blocks	Disk space hard limit.
diskspace_soft	int	0	1K-blocks	Disk space soft limit.
diskinodes	int	0	inodes	The total number of disk inodes (files, directories, symbolic links).

diskinodes_soft	int	0	inodes	The total number of disk inodes (files, directories, symbolic links). The Container is allowed to temporarily exceed the soft limit during the grace period defined by the 'quotatime' parameter.
diskinodes_hard	int	0	inodes	The total number of disk inodes (files, directories, symbolic links). The Container can never exceed this limit.
quotaugidlimit	int	0	pcs	The number of user/group IDs allowed for Container internal disk quota. If set to 0, the UID/GID quota will not be enabled. You can configure the UID/GID quota for Containers with the disabled UID/GID quota only if they are stopped.
quotaugidlimit_hard	int	0	pcs	The maximal number of user/group IDs allowed for Container internal disk quota.
counter_disk_used	int	0	bytes	The amount of disk space in use (in bytes).
counter_disk_share_used	float	0	percent	The ratio of the real disk space consumption by the Container against the disk space limit set for this Container.
counter_disk_limit	int	0	bytes	The total amount of disk space that can be consumed by the Container.

Load average counters (counters_vz_loadavg class)

Counter ID	Value	Type	Units	Description
counter_loadavg_l1	float	0	pcs	The average number of processes in the kernel run queue for the last minute.

counter_loadavg_l2	float	0	pcs	The average number of processes in the kernel run queue for the last 5 minutes.
counter_loadavg_l3	float	0	pcs	The average number of processes in the kernel run queue for the last 15 minutes.

System info counters (counters_vz_system class)

Counter ID	Value	Type	Units	Description
counter_system_users	int	0	number	Number of users.
counter_system_uptime	int	1	seconds	The time elapsed since the last server startup.

SLM counters (counters_vz_slm class)

Counter ID	Value	Type	Units	Description
slmmemorylimit	int	0	bytes	The total amount of memory that can be consumed by the Container.

Memory counters (counters_vz_memory class)

Counter ID	Value	Type	Units	Description
counter_memory_used	int	0	bytes	The total amount of memory used by the Container.
counter_memory_share_used	float	0	percent	The ratio of the real physical memory usage of the Container against the memory limit set for this Container, in percent.
counter_memory_limit	int	0	bytes	The total amount of memory that can be allocated to the Container.

Network counters (counters_vz_hw_net class)

Counter ID	Value	Type	Units	Description
counter_net_incoming_bytes	int	2	bytes	The amount of incoming network traffic in bytes.
ounter_net_incoming_packets	int	2	pcs	The amount of incoming network traffic in packets.

counter_net_outgoing_bytes	int	2	bytes	The amount of outgoing network traffic in bytes.
counter_net_outgoing_packets	int	2	pcs	The amount of outgoing network traffic in packets.

Hardware Node CPU counters (counters_cpu class)

Counter ID	Value	Type	Units	Description
counter_cpu_system	int	2	seconds	System CPU time.
counter_cpu_user	int	2	seconds	User CPU time.
counter_cpu_nice	int	2	seconds	Nice CPU time.
counter_cpu_idle	int	2	seconds	Idle CPU time.
counter_cpu_system_states	int	0	percent	System CPU time in percent.
counter_cpu_user_states	int	0	percent	User CPU time percent.
counter_cpu_nice_states	int	0	percent	Nice CPU time in percent.
counter_cpu_idle_states	int	0	percent	Idle CPU time in percent.
counter_cpu_used	float	0	percent	CPU usage in percent.
counter_cpu_share_used	float	0	percent	The ratio of CPU time consumed by the server to current limit.
counter_cpu_limit	float	0	percent	CPU limit of the share the server will get.

Hardware Node disk counters (counters_disk class)

Counter ID	Value	Type	Units	Description
counter_disk_space_used	int	0	1K-blocks	Disk space used.
counter_disk_space_free	int	0	1K-blocks	Disk space free.
counter_disk_inodes_used	int	0	inodes	Disk inodes used.
counter_disk_inodes_free	int	0	inodes	Disk inodes free.
counter_disk_used	int	0	bytes	Disk space used in bytes.
counter_disk_share_used	float	0	percent	The ratio of used disk space to current limit.
counter_disk_limit	int	0	bytes	Total disk space available for the server.

Hardware Node memory counters (counters_memory class)

Counter ID	Value	Type	Units	Description
counter_memory_mem_used	int	0	bytes	Amount of used memory.
counter_memory_mem_free	int	0	bytes	Amount of available free memory.
counter_memory_swap_used	int	0	bytes	Amount of used swap.
counter_memory_swap_free	int	0	bytes	Amount of available free swap space.
counter_memory_used	int	0	bytes	Memory used by the server.
counter_memory_share_used	float	0	percent	The ratio of used memory to current limit.
counter_memory_limit	int	0	bytes	Total memory available for the server.

Hardware Node network counters (counters_net class)

Counter ID	Value	Type	Units	Description
counter_net_incoming_bytes	int	2	bytes	Amount of incoming network traffic in bytes.
counter_net_incoming_packets	int	2	pcs	Amount of incoming network traffic in packets.
counter_net_outgoing_bytes	int	2	bytes	Amount of outgoing network traffic in bytes.
counter_net_outgoing_packets	int	2	pcs	Amount of outgoing network traffic in packets.

Hardware Node load average counters (counters_loadavg class)

Counter ID	Value	Type	Units	Description
counter_loadavg_l1	float	0	pcs	Average number of processes in the system run queue of kernel for the last 1 minute.
counter_loadavg_l2	float	0	pcs	Average number of processes in the system run queue of kernel for the last 5 minutes.

counter_loadavg_l3	float	0	pcs	Average number of processes in the system run queue of kernel for the last 15 minutes.

Hardware Node system info counters (counters_system class)

Counter ID	Value	Type	Units	Description
counter_system_uptime	int	1	seconds	Processor uptime.
counter_system_users	int	0	number	Number of users.

CHAPTER 7

Appendix B: States and Transitions

The following tables list the available server state and transition codes.

States

State Code	State Name	Description
0	unknown	The state is unknown or not supported.
1	non-existent	This state may be returned when a machine is destroyed or as an indication of the previous state when a machine is created.
2	config	The configuration information is available for a Virtuozzo Container but the Container's private area is missing.
3	down	The Container is down.
4	mounted	The Container is mounted.
5	suspended	The Container is suspended.
6	running	The Container is running.
7	repairing	Virtuozzo Container is being repaired. See `envm/repair (p. 215)` for more information.
8	license violation	The license is not valid.

Transitions

Transition Code	Transition Name	Description
0	none	The server is in a stable state (see table above).
1	unknown	The transition is unknown or not supported.
2	creating	The Container is being created.
3	mounting	The `vzaenvm/mount` operation (p. 590) is in progress.
4	starting	The Container is starting.
5	stopping	The Container is stopping.
6	unmounting	The `vzaenvm/umount` operation (p. 598) is in progress.
7	destroying	The server is being destroyed.

8	moving	The `relocator/move` operation (p. 397) is in progress.
9	cloning	The `relocator/clone` operation (p. 398) is in progress.
10	migrating	One of the `migration operations` (p. 378) is in progress.
11	starting-repair	The beginning of the `envm/repair (p. 215)` operation (the Container is in transition to the "repairing" state).
12	stopping-repair	The ending of the `envm/repair (p. 215)` operation (the Container is being brought back to the "running" state).
13	setting	The `envm/set` operation (p. 217) is in progress.
14	updating	The `vzaenvm/upgrade` (p. 599) operation is in progress.
15	backing-up	The `backup_env` operation (p. 117) is in progress.
16	restoring	The `restore_env` operation (p. 121) is in progress.
17	reinstalling	The `vzaenvm/recover_template` operation (p. 591) is in progress.
18	suspending	The `envm/suspend` operation (p. 227) in progress.
19	resuming	The `envm/resume` operation (p. 216) is in progress.
20	restarting	The `envm/restart` operation (p. 215) is in progress.

CHAPTER 8

Appendix C: Error Codes

System

0 Unknown error. Happens when some internal error cannot be translated into Agent errors

1 Internal error

2 Target not found

3 Cannot get authorization information for user

Utility errors

100 Cannot save ssh key

101 Cannot connect to agent

Packets

400 Invalid (depending on protocol) packet

401 Invalid (depending on protocol) packet

402 Virtuozzo is not functional or improperly installed

403 Container is not functional or improperly installed

404 System errors

405 Error opening file

406 Error reading file

407 Error writing file

408 Error creating directory

409 Error getting information about file

410 VEID 0 cannot be specified here

411 Error reading socket

412 Error writing socket

413 VEID is absent in packet

414 Unauthorized access to Container

415 Error renaming file to

416 Error executing cmd

417 The specified Container doesn't exist

418 The specified Container is not running

419 The specified Container is not mounted

420 Can't get list of Containers

421 Unknown host ID is specified

422 Cannot initialize connection to remote host

423 Node doesn't have master to allocate resource

424 The function is not supported in this version

425 Distribute agent to failed

426 Agent uninstall failed

427 Only VEID 0 can be specified here

428 Invalid environment was specified

userm

501 uid is already in use

502 gid is already in use

503 Specified user/group doesn't exist

504 Can't remove user's primary group

505 User name is already in use

506 Group name is already in use

507 Can't read/update group file

508 Can't read/update passwd file

509 User to modify is logged in

510 Insufficient space to move home directory

511 Invalid user/group name

512 Specified Container is not found

513 Can't create/move/remove home directory

514 Authentication error

523 Can't assign Container list to Service Container user

530 No memory

550 Container isn't run

551 No free uid/gid in given range

552 Some file is brocken

599 Error in user/group management

600 There is not enough free space in Container to write into passwd file

601 Can't get stat information for /etc/passwd file

602 Can't get stat file system information for specified Container

servicem

701 Can not get list of services

702 Can not set service levels

703 Can not start/stop/restart service

704 Can not get service status

705 Can not get levels of service

706 Unknown service name

707 Xinetd is stopped

710 Can not init service management

firewallm

900 Internal firewall management error

901 Container is not running

902 Can't get Container Root

903 Cannot add the rule from config

904 Error setting the default DROP policy for Container

905 Error setting the default ACCEPT policy for Container

906 Error getting the default policy for Container

907 Error getting the active protocols' list for Container

908 Invalid rule parameters were passed

909 Specified rule of Container was not found

910 Error flushing the firewall of Container

911 Error parsing the config file of Container

912 Error writing the config file of Container

913 Error setting up the firewall chains for Container

914 Error saving the iptables information for Container

vem

1000 Error reading UBC values

1001 Error reading Container configuration file

1004 Error invoking vzctl utility

1005 Config error of Container

1006 Private area of Container already exists

1007 Error repairing Container

1008 Error moving Container

1009 Error cloning Container

1010 The Container is in repaired state Stop repairing first

1011 This operation is prohibited for the Service Container

1012 Invalid hostname

1013 Error setting ugid quota

1014 Error getting ugid quota

1016 Some other operation is done with the Container at the moment

1017 Quota file is corrupted

1018 ugid quota not accounted

1019 get/set protocol respond message failed

1020 Error initialize VZ parameters validation

1021 Error parametrs validation

1022 Error in VEID allocation in pool

1023 Error in IP allocation in pool

1024 Container/IP pool is not accessible

1025 Error to modify Container parameters

1026 Error Container is already running

1027 Error Container is not running

1028 Error IP address already used

1029 Error IP address not configured

1030 Can't create Container

1031 Can't upgrade Container

1032 Can't get scripts

1033 Can't set scripts

1034 Can't del scripts

tem

1299 Unknown error

1201 Can not get upload directory path

1202 Can not get list of templates

1203 Can not generate obsoleted rpm

1204 Can not get list of dependencies

1205 Can not consruct path to config file

1206 Can not open template config file

1208 Can not write to config file

1209 Can not install template rpm

1210 Can not get name of template rpm

1211 Can not uninstall template rpm

1212 Can not create template

1213 Can not add template in Container

1214 Can not remove template from Container

1215 Config file is invalid

1216 Removing these package would break dependencies in Container(s)

1217 run vzpkgcache error

1218 Template(s) already installed

1219 Failed template dependencies

1220 Use --force

1221 Can not get distributions

1222 Can not migrate template

computerm

1301 Can not open VZ vocabulary

1302 Can not parse VZ vocabulary

1303 No such category in VZ vocabulary

1304 No such parameter in VZ vocabulary

1311 Can not write Container/VZ config

1312 Can not delete sample Container config

1313 Can not rename sample Container config

1314 Can't get partition info

1315 Can't get memory info

1316 Can't get CPU info

1317 Can't get per interface info

1318 Can not get upload directory path

1320 Can not migrate Container(s)

1321 Can not set key

1322 Can not get key

1323 Can not get log list

1324 Can not get log

1325 Can not set VZ config

1326 Can not get traffic

1327 Can not set network parameters

1328 Can not get network parameters

1329 Migrate object already exists

1333 Can not split hardware node

1334 Configuration file already exists

1335 Can not get OS info

1336 Can't get redirect services

1337 Can't reconfigure redirect services

1338 Can't set the date

1339 Can't get the date

1340 Can't get the time zones information

dbm

1401 No disk space for create log file

1402 Query not complete

1403 No memory

1404 Unknown parameter

1405 Excessive periods in query range

1406 Invalid period

1407 Can't read counters

devm

1501 Can not get list of mounts

1502 Can not get device info

1503 Mount entry already exists

1504 Can't add/delete permanent mount point

1505 Can't mount/umount device

1506 Can't create mount point

1507 Container is in improper state

1508 Can't create/delete/resize drive \n

1509 Can't forward/list/remove forward on device \n

licensem

1602 Can not instal new license

1603 Can not get license information

1604 Can not get HWID information

1605 Can not remove license

1606 Can not activate license

1607 Can not update license

1608 KA server is not available

Periodic Collectors

1701 Monitor doesn't exist

1702 Quota accounting for veid is off

Mail Configuration

1801 Can't get user info (home dir)

1802 Can't read mail template dir

1803 Error with template file

processm

1901 Error getting processes info for Container

1902 Error parsing processes info for Container

1903 Error sending signal to process

1904 No such pid

1905 Can't get ID for pid

1906 Error invoking vzctl utility

3600 Error run program

File Manager

2001 Wrong credentials

2002 Can't call function

2050 Resource exceeded

2051 Access denied

2052 File busy

2053 Input/Output error

2054 No space on disk

2055 Path already exist

2056 Bad (broken, doesn't exist) path

2057 User/Group name not found (provided for 'chown' command)

Mailer Operator

2301 No mail services in Container

2302 This forward leads to loop

2303 Error while synching

2304 Mail box already exist

2305 No such mail box

2306 Domain already exist

2307 No such domain box

2308 Mail list already exist

2309 No such mail list

2310 Can't create/remove domain directories

2311 Email is invalid

2312 Can't save config

2313 Can't load config

2314 Domain has users

Backup management

2501 Backup with such id is not found

2502 Can't backup Container

2503 Can't restore Container from backup

2504 Can't set vzbackup config

2505 Backup number exceeded current limit

2506 Can't get vzbackup config

2507 Snapshot wasn't created

tbs

2601 Can't read sshd config

2610 Can't check network parameters

2615 Can't fix network parameters

2620 Can't check disk quota parameters

2625 Can't fix disk quota parameters

2630 Can't check VZ filesystem

2640 Can't template check

2645 Can't fix template

2646 Can't set or get support channel status

sessionm

2701 No such session is found

2702 IP doesn't match the session

2703 Deprecated Error 2703 "Session expired"

2704 Cannot authenticate user due to a system error

2705 Authentication failure - either user name or password is incorrect

2706 Cannot distribute session

2707 Cannot duplicate session

2708 Cannot logout permanent session

Global resource management

2802 DB Error

2803 VEID pool exhausted

2804 IP pool exhausted

2805 Bad VEID in parameters already exist in pool not in pool not found in pool

2806 Bad IP in parameters already exist in pool not in pool not found in pool

2807 Node is not master, so it cant support called command allocate release

2901 DB Error

2902 Invalid pool (not in user range)

2903 Master initialization error

2904 Client initialization error

2905 Can't remove client

2906 Can't remove master

2907 Can't upload credentials

2908 Client already has some master

2909 Error

2910 Node is not master

SQL operator

3101 DB error

3111 Unknown field in select part of packet

3151 Error parsing (at this moment) where

3152 Error evaluating (at this moment) where

authm

3201 User/role with name already exists

3202 Authentication error

3203 Login is disabled

3204 User/role is not found

3205 System error

3206 Can't delete the role -- other users/roles depend on it

packagem, vzapackagem

3501 Fatal error installing a package

3502 Error removing a package

3503 Error updating a package

3504 Error listing a package

3505 Error getting package information

3506 Error cleaning package cache

3507 Error migrating a package

3508 Error fetching a package

3509 Error forcing package installation

3510 Package already installed

3511 Invalid package distribution

envm, vzaenvm

3800 Error reading Environment configuration

3801 Some other operation is done with Environment at the moment

3802 Error invoking external utility

3803 Configuration error

3804 This operation is prohibited for this Environment

3805 Error writing Environment configuration

Scheduler

4000 Value for time is our of range

4001 Task not found

System errors

-2 The request was timed out

-3 The request was cancelled

-4 Connection to vzagent was closed

-5 Can't parse vzagent message

-6 Invalid parameters was specified

-7 Invalid vocabulary was specified or vocabulary is missing

-8 The request was not properly initialized

-9 Library was unable to connect for the first time

-10 Library tried to reconnect N times but did not succeeded

-11 Internal error

-12 Invalid environment was specified

-13 Low memory allocation failed

-14 State of the object prohibits requested call

-15 Out of system resources (HANDLEs descriptors)

-16 Invalid connection passed

-17 No requested data available

filer

-1101 Can't open local file

-1102 Can't read from local file

-1103 Can't write to local file

1104 Local file exists

SQL errors

-1201 System error in database

-1202 VEPool exhausted

-1203 IPPool exhausted

-1204 Bad veid

-1205 Bad IP

-1250 For internal use

Other

-424 The function is not supported in this version

Index

A

About This Guide • 5
aceType • 26
add • 291, 428, 435
add_group • 73, 480
add_realm • 76
add_resource • 410
add_to_group • 78
add_user • 80, 474
Adding a VLAN adapter to the Hardware Node. • 294
Adding and removing an IP address to/from a VLAN adapter. • 304
Agent Messages • 16
alert_dataType • 27
alertm • 62
allocate • 415
Appendix A
Performance Counters • 635
Appendix B
States and Transitions • 644
Appendix C
Error Codes • 646
Assigning a Virtuozzo virtual network ID to a physical network adapter or to a non-Virtuozzo virtual network. • 305
Attaching/detaching LAN/VLAN adapters to/from a network bridge. • 303
audit_event_dataType • 229
auth_nameType • 28
authenticate • 82, 492
authm • 71

B

backup_dataType • 105
backup_env • 115
backup_options_baseType • 105
backup_optionsType • 106
backup_storagem • 104
backupid_type • 108
backupm • 105
backupm_configType • 108
backupType • 109
Base Types and Interfaces • 23

C

calc_env_config • 387
Calls • 63, 73, 104, 114, 140, 156, 160, 187, 192, 229, 237, 262, 269, 283, 290, 308, 316, 348, 359, 368, 376, 396, 410, 428, 435, 451, 460, 474, 505, 568, 571, 595, 597, 601, 609, 613, 617, 631, 634
cancel • 506
chainType • 260
chmod • 251
chown • 252
class_rateType • 595
classType • 347, 395
clean • 340
clone • 392, 634
clone_optionsType • 373, 593, 633
close • 507
Common Types • 23, 557, 623
common_deviceType • 159
Complex Types • 26, 558, 623
compress_perf_dataType • 348
computerm • 139
configType • 433
configuration • 508
configurationType • 611
connect • 515
connection_infoType • 29
connectivity_infoType • 29
copy • 243
count_registered • 471, 517
cpu_loadType • 30
cpuType • 30
create • 193
create_drive • 161
Creating a network bridge on the Hardware Node. • 292
credentialType • 31
credType • 233

D

daily_triggerType • 419
data_storagem • 154
datetime_type • 24
del • 306, 438
del_from_group • 84
del_group • 85, 479

del_realm • 86
del_script • 572
del_user • 87, 477
delete • 264, 279
delete_drive • 165
destroy • 199, 446
determine_env • 573
deviceType • 623
devm • 157
dir_realmType • 71
disable • 267
diskType • 139
distribute • 520
Documentation Conventions • 7
download • 249
ds_configurationType • 155
ds_locationType • 155
ds_object_infoType • 155
ds_object_path_type • 156
duplicate_session • 462

E

edit_group • 88, 488
edit_user • 90, 478
eid_listType • 31
eid_type • 25
Elements • 496
enable • 266
env_backup_dataType • 110
env_config_event • 498
env_config_event_dataType • 495
env_configType • 31
env_resourceType • 32
env_samplem • 187
env_sampleType • 33
env_security_objectType • 33
env_status_event • 496
env_status_event_dataType • 494
env_statusType • 33
envm • 192
envType • 34
errorType • 36
event_dataType • 34
event_log • 229
Events • 494
eventType • 35
exclude_listType • 111
execute • 368

F

Feedback • 10
fetch • 342
file_listType • 36
filer • 233
fileType • 235
firewallm • 259
format_drive • 166
forward_device • 167

G

General Conventions • 9
generate_pass • 522
get • 262, 354, 364, 452, 467, 606
get_actual • 358
get_alerts • 63
get_auth_name • 92
get_channel_status • 619
get_config • 137, 447, 599, 614
get_configuration • 523
get_date • 151
get_disk • 140
get_eid • 531
get_env_info_optionsType • 110
get_env_voc • 449
get_events • 229
get_group • 93, 486
get_history • 358
get_hwid • 280
get_info • 134, 169, 200, 337, 441
get_info_optionsType • 112
get_key_status • 620
get_limits • 490
get_list • 206, 438
get_log • 208, 396
get_log_info • 401
get_mail_template • 284
get_mounts • 170
get_native_config • 226
get_network • 145
get_ops • 308
get_plugins • 531
get_pool • 414
get_realm • 97, 535
get_relay • 287
get_sample_conf • 187
get_script • 574
get_sid • 99
get_split_conf • 575
get_state • 537
get_storage_config • 156
get_system • 143
get_top • 403
get_ugid_quota • 578
get_user • 100, 482
get_version • 540
get_virtual_config • 228
get_vocabulary • 449, 541
get_vt_info • 209

get_vt_settings • 210
get_zones_info • 153
group_add_user • 484
group_del_user • 485
group_set_users • 489
groupType • 37
guest_os_memory • 625
guid_type • 25

H

How to Use This Guide • 7
hw_notesType • 376, 594

I

infoType • 37
install • 272, 317, 615
install_tools • 631
Interfaces • 62, 568, 630
interfaceType • 40
intervalType • 41
ip_addressType • 41
ip_poolType • 42
ip_rangeType • 42
ip_type • 25
is_enabled • 265

K

kill • 371, 609

L

license_eventType • 268
licensem • 267
licensesType • 268
licenseType • 268
link • 253
list • 127, 237, 269, 295, 331, 430, 614
list_device • 174
list_forward • 176
list_optionsType • 112
list_sessions • 468
load_avg_statsType • 43
load_avgType • 44
log_options_baseType • 44
log_optionsType • 44, 558
login • 543
login_as • 461
logout • 461
logType • 395

M

mail_template_list • 283
mailer • 282
Message Body • 21
Message Header • 16
migrate • 345
migrate_p2v • 377
migrate_v2p • 386
migrate_v2v • 383
mkdir • 245
mode_type • 237
modType • 45
monthly_day_of_week_triggerType • 421
monthly_triggerType • 423
mount • 579
mount_deviceType • 158
move • 246, 391

N

named_listType • 45
native_configType • 46
navigate_wildType • 235
navigateType • 234
net_addressType • 46
net_bridgeType • 290
net_classType • 47
net_configType • 596
net_deviceType • 47
net_nicType • 48
net_vethType • 559
net_vlanType • 290
networkm • 289
networkType • 434
new_mount • 178

O

once_triggerType • 424
op_log • 308
operator_functionalType • 48
Organization of This Guide • 6
osType • 49

P

p2v_migrate_optionsType • 373
package_debType • 313
package_linuxType • 313
package_rpmType • 313
package_std_vztemplateType • 561
package_vztemplateType • 561
packagem • 312
packagesType • 314
packageType • 49
Parallels Server Types and Interfaces • 623
parameterType • 267
parse • 276
partitionType • 139
pause • 631
perf_dataType • 50
perf_mon • 347

perf_statType • 51
ping • 545
pkg_cmdType • 314
pkg_paramsType • 315
pkg_setup_cmdType • 315
policy_ruleType • 51
policyType • 260
port_rangeType • 261
post • 288
Preface • 5
Primitive Types • 23
privilegeType • 25
problem_report • 621
proc_info • 359
processesType • 51
processm • 367
progress_eventType • 555
ps_infoType • 52
put • 465
put_ops • 311

Q

qosType • 52
quota_limit • 570
quota_type • 570

R

readlink • 256
realmType • 53
reboot • 150
recover_template • 580
redirect_serviceType • 562
Reference Format • 11
reg • 632
register_client • 470, 547
release • 418
relocator • 373
remove • 130, 242, 322, 430
remove_forward • 182
remove_key • 621
remove_mail_template • 286
remove_optionsType • 113
remove_resource • 411
repair • 213
res_log • 395
resize_drive • 183
resource_alert • 501
resource_alertType • 62, 495
resource_ip_poolType • 409
resource_ipType • 410
resource_poolType • 408
resourcem • 408
resourceType • 54, 408
restart • 214, 457
restore_env • 119
restore_optionsType • 113
resume • 214, 582
ruleType • 261

S

sample_confType • 54
scheduler • 419
scsi_deviceType • 160
search • 132, 258
search_optionsType • 114
security_descriptorType • 55
security_objectType • 55
security_principalType • 72
selective_restore_env • 123
selective_restore_optionsType • 114
server_group • 433
server_group_alertType • 63
service_actionType • 450
servicem • 450
serviceType • 451
sessionm • 459
sessionType • 460
set • 215, 263, 301, 455
set_config • 135, 447, 597, 614
set_date • 152
set_key • 620
set_log • 399
set_mail_template • 285
set_pool • 412
set_realm • 103
set_relay • 287
set_script • 583
set_startup_type • 458
set_storage_config • 157
set_ugid_quota • 584
set_user_password • 585
set_vt_settings • 221
Shell Prompts in Command Examples • 9
sidType • 26
Simple Types • 24, 557
sp_modType • 72
start • 222, 456
start_channel • 617
start_monitor • 349, 359, 602
stat • 254
statsType • 56
stop • 223, 456
stop_channel • 618
stop_monitor • 353, 364, 606
stop_repair • 224
subscribe • 548
subscribe_alert • 67
suspend • 225, 586

sys_infoType • 56
system • 504
System Interface and Special Packets • 504
system_nodeType • 57
systemType • 140

T

taskType • 424
templateType • 562
The progress packet • 553
tokenType • 57
transferType • 58
transport_type • 26
triggerType • 426
Types • 62, 71, 105, 139, 155, 158, 229, 233, 260, 267, 283, 290, 313, 347, 373, 395, 408, 419, 433, 450, 460, 494, 504, 555, 569, 593, 595, 611, 633
Typographical Conventions • 8

U

ugid_quota_info • 569
umount • 184, 587
uninstall • 552, 616
unreg • 632
unsubscribe • 552
unsubscribe_alert • 69
update • 281, 326, 432
update_device • 632
updateType • 613
upgrade • 588
upload • 247
usageType • 59
userm • 474
userType • 59

V

v2p_migrate_optionsType • 376
v2v_migrate_optionsType • 375
validate • 589
validationType • 570
veid_type • 558
venv_configType • 60, 563, 625
verify • 465
Virtuozzo Containers Types and Interfaces • 557
virtuozzo_configType • 566
voc_idType • 434
voc_parameterType • 60, 504
vocabularyType • 61
vt_infoType • 61, 567
vt_settingsType • 61, 567, 629
vzadevm • 568
vzaenvm • 569
vzanetworkm • 595
vzapackagem • 601
vzaproc_info • 601
vzaprocessm • 609
vzarelocator • 593
vzasupport • 616
vzaup2date • 610
vzpenvm • 631
vzprelocator • 633

W

weekly_triggerType • 427
Who Should Read This Guide • 5
windows_deviceType • 159

X

XML Code Samples • 15
XML Message Specifications • 11