
Parallels

Parallels Agent

Porting VZAgent 3.0.x applications to Parallels
Agent 4.0.0

1.0



(c) 1999-2008

ISBN: N/A
Parallels
13755 Sunrise Valley Drive
Suite 600
Herndon, VA 20171
USA
Tel: +1 (703) 815 5670
Fax: +1 (703) 815 5675

© 1999-2008 Parallels. All rights reserved.
Distribution of this work or derivative of this work in any form is prohibited unless prior written permission is
obtained from the copyright holder.

Contents

Preface	4
About This Document.....	4
Porting XML Applications	5
Protocol Version	6
Identifying Hardware Nodes and VPSs	7
Plug-in Architecture, OOP, Inheritance	9
Virtuozzo VE Configuration.....	10
Virtuozzo VE Management	11
Hardware Node Management	11
Performance Monitoring and Statistics Collection	12
Events and Alerts.....	12
File Transfer and Temporary Upload Directory	13
Authentication Management.....	13
Miscellaneous	14
Index	15

CHAPTER 1

Preface

In This Chapter

About This Document	4
---------------------------	---

About This Document

The Parallels Agent 4.0.0 XML API is a complete rewrite of the VZAgent 3.0.x API. The new protocol is not backwards compatible with the older one. This guide discusses the key differences between the two protocol versions and provides information on how to port the existing VZAgent 3.0.x applications to Parallels Agent 4.0.0.

CHAPTER 2

Porting XML Applications

This chapter describes how to port the existing VZAgent applications to Parallels Agent 4.0.0. VZAgent 3.0.3 applications will continue to work with Parallels Agent 4.0.0 without any changes. Earlier protocol versions are not officially supported so the existing applications using protocol version earlier than 3.0.3 may not work correctly and therefore should be either upgraded to version 3.0.3 or ported to version 4.0.0. To upgrade an existing application to version 3.0.3, all you have to do is to change the protocol version number in the headers of your XML messages. To port the existing applications to version 4.0.0, you should read this guide and follow the instructions provided.

If you are going to be using a version 3.0.3 application with Parallels Virtuozzo Containers 4.0, you should be aware that some of the old VZAgent functionality will not work with it. The following list describes the interfaces that are no longer supported:

- The `snapshotm` interface is not supported. In the previous version, the interface allowed to create and manage Virtuozzo OS templates in the EZ Template format which was introduced in Virtuozzo 3.0. This functionality has been moved to the `pkgm` interface in Parallels Agent 4.0.0.
- The `mailc` interface is not supported. The interface allowed to manage e-mail templates used for automatic system alert notifications. The format of the templates has changed, some of the variables have changed as well. This functionality has been moved to the new `mailer` interface.
- The `nbh` interface is no longer supported. The name-based hosting functionality has been dropped in Virtuozzo Containers 4.0.
- The `global` interface is not supported. The interface allowed to set up and manage a Virtuozzo group (formerly Virtuozzo cluster) and to perform certain Hardware Node related tasks. This functionality has been moved to the new `clusterm` interface in VZAgent 4.0.0.
- The support of managing VE ID pools for the Hardware Nodes has been dropped. In Virtuozzo 4.0, you should manually assign IDs to your VEs during their creation.
- In addition, to provide the compatibility with VZAgent 3.0.3 API, you have to assign a public IP address to the Service VE and set the password for the `vzagent0` user inside this VE using the `vzctl set` command:

```
# vzctl set 1 --ipadd public_IP_address --userpasswd vzagent0:*****
```

In This Chapter

Protocol Version	6
Identifying Hardware Nodes and VPSs	7
Plug-in Architecture, OOP, Inheritance.....	9
Virtuozzo VE Configuration.....	10
Virtuozzo VE Management	11
Hardware Node Management	11
Performance Monitoring and Statistics Collection	12
Events and Alerts	12
File Transfer and Temporary Upload Directory	13
Authentication Management.....	13
Miscellaneous	14

Protocol Version

The current VZAgent protocol version is 4.0.0. The first thing that you have to do is to update your existing code so that the outgoing XML packets have the correct protocol version. The protocol version is specified using the `version` attribute of the `packet` element, which is the root element of every VZAgent XML message.

Old version:

```
<packet version="3.0.3">
```

New version:

```
<packet version="4.0.0">
```

Identifying Hardware Nodes and VPSs

Prior to VZAgent v.4.0.0, a Hardware Node was identified just like any other computer on a network -- by IP address or hostname. A VPS was identified by an internal Virtuozzo ID, which was assigned to every VPS at the time of creation. The Hardware Node and VPSs were two completely different and separate entities. In VZAgent 4.0.0, every participating machine (physical or virtual) is automatically assigned a universally unique ID (called *Environment ID* or *EID* for short) and is identified within VZAgent infrastructure by this ID alone. Every Hardware Node has an EID which is assigned to it as soon as VZAgent is installed on it. Every Virtuozzo VE (formerly VPS) also has an EID, which is assigned to it at the time the VE is created. In VZAgent 4.0.0 it doesn't matter whether a machine is a physical server (Hardware Node) or a virtual server (VE) -- all that is needed to access a server and perform administration tasks on it is the EID.

What this means to you is that most API calls use this ID now instead of Virtuozzo internal VPS ID, so you will have to update your code accordingly. In addition, new functions were added to get the list of EIDs from a particular Hardware Node or from a Virtuozzo group (formerly Virtuozzo cluster). You will have to use these functions to get the list of the available EIDs when needed.

The following example demonstrates how a VE is started using the two VZAgent API versions. Note the `<veid>` parameter in v. 3.0.3 and the `<eid>` parameter in v. 4.0.0

VZAgent 3.0.3 and earlier

```
<packet version="3.0.3" id="611">
  <target>vem</target>
  <data>
    <vem>
      <start>
        <veid>103</veid>
      </start>
    </vem>
  </data>
</packet>
```

VZAgent 4.0.0

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

The Virtuozzo-level VPS ID (or VEID for short) still exists in the current Virtuozzo version but from the VZAgent API point of view, the significance of this ID has been reduced to the point where only a very few calls actually use it.

The other important difference is that many API calls don't have a parameter that would identify a VE the call is targeted at. This means that they have neither the old-style VEID nor the new-style EID parameter. By default, these calls are processed at the Hardware Node level and perform operations on the Node itself. In order for a call like that to perform its operation on a particular Virtuozzo VE, you will have to use the *call forwarding* feature (see the example below). In comparison, all similar calls in VZAgent v. 3.0.x always had a `veid` (VPS ID) parameter. Consider the following example:

VZAgent 3.0.0

Note the `veid` parameter specifying the VPS ID to get the list of files from.

```
<packet version="3.0.0">
  <target>filem</target>
  <data>
    <filem>
      <list>
        <veid>777</veid>
        <cwd>Lw==</cwd>
        <path>Lw==</path>
      </list>
    </filem>
  </data>
</packet>
```

VZAgent 4.0.0

The same call in VZAgent 4.0.0 doesn't have the `veid` parameter. To get the same results as in the call above, we use the `dst/host` element in the message header to forward the call to the Virtuozzo VE specified by its VEID. Without call forwarding we would receive the list of files from the Hardware Node that we are currently connected to. The calls like that will have to be identified in your existing code and the necessary changes will have to be made.

```
<packet version="4.0.0">
  <dst>
    <host>24b9acf5-8ca5-49c9-b7b1-4c93fe048389</host>
  </dst>
  <target>filer</target>
  <data>
    <filer>
      <list>
        <cwd>Lw==</cwd>
        <path>Lw==</path>
      </list>
    </filer>
  </data>
</packet>
```

Plug-in Architecture, OOP, Inheritance

A plug-in architecture has been introduced in VZAgent 4.0.0. The core VZAgent components provide the base functionality common to all possible server management types and scenarios. Plug-in modules are extensions that provide additional functions that are specific to a particular server management area or technology. The examples of plug-in modules include Authentication Engine and Virtuozzo VE management. Other plug-ins may be developed in the near future by SWsoft or by third party developers.

In some cases plug-in modules inherit their functions from the base VZAgent components. For example, Virtuozzo plug-in inherits many of its interfaces and data types from the base interfaces and data types. The descendant objects extend the base objects by adding additional components such as API calls and parameters specific to the functionality that a plug-in module implements. In VZAgent XML API, not only the base types can be used in descendant calls but their subtypes as well, so you will have to use the correct type if you want to achieve a desired result.

What this means to you is that some of the API calls in your application will no longer be valid because the functionality has been moved to a plug-in module. Calls like that will have to be identified and will have to be revised. In some cases, only the minor changes will be needed, like renaming the call itself and changing a few parameters. In others, you will have to rework the call by making sure that the correct data types are used and that the parameters that use these types are properly populated.

When processing result sets, the inheritance issues must also be considered in certain cases. The reason is that based on the type of the information returned, different instances of the same element in an XML packet may use different subtypes of the same base type. For example, when retrieving the system event information, the data element in the returned structure will contain the event or alert type-specific data. Depending on the type of the event or alert, the data type of the underlying event_data element will be one of the descendants of event_dataType. 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. None of that existed in the VZAgent 3.0.x API.

The functional areas that will have to be revised in this respect are as follows:

- Creating Virtuozzo VEs.
- Getting and setting the VE configuration information.
- Package management (standard and EZ templates).
- OS processes management.
- Troubleshooting, problem reporting, support management.
- Virtuozzo VE migration.
- Alerts and Events.

Virtuozzo VE Configuration

Prior to VZAgent version 4.0.0, there was a single type that was used to hold the VPS configuration information. That type was `ve_configType`. This has changed significantly.

Currently, there's a base type called `env_configType` (as you can see, the name is different). When compared to the old-style `ve_configType`, the new type is missing all of the Virtuozzo VE-specific parameters such as a list of QoS parameters for example. These parameters are now defined in a subtype called `venv_configType`, which extends the base type. When creating or editing a Virtuozzo VE, or when retrieving the configuration information for an existing VE, the `venv_configType` must be used. This means that the XML element at the root of the `config` structure must have the attribute specifying the data type being used, as shown in the following example:

```
<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:virtual_config>
```

The namespaces used in the example above must be defined in the XML message header. For example, the `venv_configType` type belongs to the `http://www.swsoft.com/webservices/vza/4.0.0/vzatypes` namespace. You must include this information in the message header before you can use the type. The following is an example of a namespace declaration:

```
<packet
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">
```

If you look closer at the new `venv_configType` structure and compare it to the old `env_configType`, you'll also see that some of the parameters are no longer there. For example, such parameters as `class_id` (Virtuozzo license class), the `nbh`-related parameters, `vzcache`, and others are now obsolete and therefore have been removed from the type definition. Many parameters, however, have not changed and are used exactly as they were in the previous VZAgent version.

Some other changes to the configuration structure will require an additional effort on your part when porting existing applications. Because of the new features that have been added to Virtuozzo networking, the network interfaces are now specified in a completely different manner. In the previous version, the network interface was defined using the single `interface` element which contained the interface name, the network class ID, and the bandwidth rate. Now, to specify the network interface information, you must use the new `net_device` element that has more parameters and is designed to handle network interfaces of different types, including the new Virtuozzo virtual networks. So, if your existing applications handle the network interface parameters, IP addresses, and other network-related settings, all of this will have to be reprogrammed. The changes are not drastic. Once you have a grasp on the basic building blocks, you will be able to make the necessary changes in minutes.

Virtuozzo VE Management

Virtuozzo VE management hasn't changed a lot in general but you will still have to revise your existing code because the interface name, the list of the available calls and the input parameters have changed.

The old `vem` interface (VPS management) was replaced with the new `vzaenvm` interface. Just as before, to create a VE you use the `create` call. If you compare the v. 3.0.3 and v. 4.0.0 of the call, you will see that they look very similar. The changes include replacing the interface name, the `veid` (VPS ID) parameter must be replaced with `eid` (the Environment ID, we discussed it earlier in this document). The new configuration structure is used in v. 4.0.0 (we already discussed it too). The new version of the call is now cleaner and easier to understand. Some additional features were added (the features are optional so not using them will not break your existing code). Other than that, the base principles of creating a Virtuozzo VE remain essentially the same.

Some of the VE management calls have the same names as before, some were renamed, and some were removed from the interface. The calls with the same names will require minor modifications, in most cases just changing the `veid` to `eid` and passing the correct `eid` value. The renamed calls must be revised in the same manner -- changing the names of the calls and verifying that the input parameters and their values. The removed calls are now obsolete.

Hardware Node Management

The old `hwm` interface no longer exists in VZAgent v. 4.0.0. The functionality that it provided has been moved to other interfaces, namely `envm` -- the Environment Management interface; `migrator/vzamigrator` -- the VE migration management; `env_samplem` -- the sample VE configuration management. The names of the most calls have changed as well. Some of the old `hwm` calls were dropped as obsolete. For example `get_ssh_key` and `set_ssh_key` are no longer used because SSH protocol is no longer supported in the current VZAgent version (TCP and SSL are the protocols used).

If you are using the `hwm` functionality in your applications, you will have to find the equivalents of the calls in the new set of API interfaces and will have to rewrite your existing code to use those interfaces instead. Individual calls will most likely have to be revised as well but the required changes should be minimal.

Performance Monitoring and Statistics Collection

In VZAgent version 3.0.x, we had "Periodic Operators" which were used to collect performance data on a periodic basis. Each system resource type had a matching "operator" that provided functionality for the resource monitoring. There was the `hw_cpu` operator to monitor the CPU utilization; the `hw_net` operator allowed to monitor the network, and so forth. None of these exist anymore. The old-style performance monitoring has been removed from the API. The all-new `perf_mon` interface has been introduced in VZAgent 4.0.0, which now provides the performance monitoring functionality. The main difference between the old-style Periodic Operators and the new `perf_mon` interface is that `perf_mon` can handle all of the available types of the resources. It does that by utilizing the new concept of performance classes, counters, and instances that has been introduced in VZAgent 4.0.0. You simply pass to `perf_mon` the appropriate class/counter/instance values and it will monitor the corresponding system resource for you.

What it means to you is that if you have an application that monitors system resources, it will have to be re-developed. You will have to add additional code that will retrieve the available performance classes and counters (although, this information can probably be retrieved just once and then hard-coded, which is entirely up to you). You will also have to use the new interface and the calls that it provides. The monitoring results are returned using new structures as well, so if you have a parser of some sort that processes the results and generates a report from them, this will also have to be changed.

Events and Alerts

Events and alerts work are handled in the same exact manner as before (you subscribe to event and alert notifications) with the following exceptions:

- Subscriptions -- the subscription names have changed.
- Result sets -- the structures of the result sets containing the event and alert information have changed significantly. Some of the parameters and their values have changed as well.

The bottom line is, if you have an existing application that monitors system for events or alerts, it will have to be revised. The subscription names that you currently use must be replaced with the new ones. The result set processing must be reprogrammed to handle the new data types, parameters, and values.

File Transfer and Temporary Upload Directory

The `file_transfer` interface and the temporary upload directory functionality are no longer available in VZAgent 4.0.0. The temp directory was used to store the output of the `dbm` (history database management) calls as text files that could later be downloaded from the directory using the `file_transfer` interface. This is no longer necessary because the new interfaces that replace the `dbm` interface (which is now obsolete) return the data directly to the client without storing it in the temp directory. The new interfaces that replace the `dbm` interface are `op_log` and `res_log`.

The temp directory was also used as an upload directory for Virtuozzo templates. In VZAgent 3.0.x, you first had to upload a template file to the temp directory using the `file_transfer/write` call and then use the `tem/set` call to install the template. The `tem` interface is now also obsolete. The new `pkgm` interface provides calls for Virtuozzo template management.

Authentication Management

VZAgent 4.0.0 introduces an entirely new user authentication management system. In VZAgent 3.0.x, user authentication was performed against the operating system user database, a Windows domain database, or the SSH user database (when an SSH connection was used). The new system uses a concept of *realms*. In VZAgent terms, realm is a user database that may reside on a local machine or on some remote network location. The type of the database can vary from the operating system user registry to an LDAP-compliant directory. A VZAgent installation can be made aware of these databases by creating realm definitions and storing them in the VZAgent configuration files. A user can then be authenticated against any of the available realms.

What it means to you is that in existing applications, the code invoking the API calls where a user ID and password is required must be revised to use the new-style login parameters and values. If that's not done, the user authentication will no longer work.

Miscellaneous

Most of the other interfaces have also changed in one way or another. In some cases, it is just the interface, the calls, and the parameters names that got changed. In others -- it's the default values, the enumerations, and the expected predefined values. Some calls may no longer exist, some were moved to a different interface, and so forth. All in all, it is safe to say that porting an existing VZAgent 3.0.x application to VZAgent 4.0.0 will require some work. Most of the effort will probably go into learning the new API concepts. The actual changes that you will have to make to your existing code should not be massive but it really depends on the complexity and size of your application. We believe that you should be able to keep the majority of your existing code by making modifications where necessary instead of re-writing the entire application from scratch.

Index

A

About This Document • 4

Authentication Management • 13

E

Events and Alerts • 12

F

File Transfer and Temporary Upload Directory
• 13

H

Hardware Node Management • 11

I

Identifying Hardware Nodes and VPSs • 7

M

Miscellaneous • 14

P

Performance Monitoring and Statistics
Collection • 12

Plug-in Architecture, OOP, Inheritance • 9

Porting XML Applications • 5

Preface • 4

Protocol Version • 6

V

Virtuozzo VE Configuration • 10

Virtuozzo VE Management • 11