Oracle® VM

Manager Command Line Interface User's Guide for Release 3.3

E50252-06
March 2016

Abstract

Document generated on: 2016-03-14 (revision: 5487)

Table of Contents

Preface

The Oracle VM Manager Command Line Interface User's Guide is your reference to the Oracle VM Manager Command Line Interface (CLI). The CLI is intended to offer the same features as the Oracle VM Manager graphical user interface, so you can manage your Oracle VM environment using the CLI and not need to use the Oracle VM Manager graphical user interface, if you so choose. The CLI connects to an Oracle VM Manager instance from either the Oracle VM Manager host, or another client computer using an ssh connection. You can script CLI commands using any programming or scripting language of your choice. A few Expect scripts are provided for your reference to get you started. This Guide gives you an overview of how to connect to the CLI, examples on how to use it to set up your environment, and a complete syntax reference.

1 Audience

This document is intended for Oracle VM administrators with privileged access to the physical and virtual resources of the Oracle VM environment. This guide assumes that you have an in depth knowledge of Oracle VM (see the *Oracle VM Manager User's Guide*), and that you are familiar with Oracle Linux system administration and Linux command line operation.

2 Related Documents

For more information, see the following documents in the Oracle VM documentation set:

- *Oracle VM Release Notes*
- *Oracle VM Installation and Upgrade Guide*
- *Oracle VM Concepts Guide*
- *Oracle VM Manager Getting Started Guide*
- *Oracle VM Manager User's Guide*
- *Oracle VM Manager Command Line Interface User's Guide*
- *Oracle VM Administrator's Guide*
- *Oracle VM Windows Paravirtual Drivers Installation Guide*
- *Oracle VM Web Services API Developer's Guide*
- *Oracle VM Security Guide*
- *Oracle VM Licensing Information User Manual*

You can also get the latest information on Oracle VM by going to the Oracle VM Web site:

http://www.oracle.com/us/technologies/virtualization/oraclevm

3 Command Syntax

Oracle Linux command syntax appears in `monospace` font. The dollar character ($), number sign (#), or percent character (%) are Oracle Linux command prompts. Do not enter them as part of the command. The following command syntax conventions are used in this guide:

Convention	**Description**
backslash \	A backslash is the Oracle Linux command continuation character. It is used in command examples that are too long to fit on a single line. Enter the command as displayed (with a backslash) or enter it on a single line without a backslash: `dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b \` `count=10000`
braces { }	Braces indicate required items: `.DEFINE {macro1}`
brackets [ ]	Brackets indicate optional items: `cvtcrt` *`termname`* `[`*`outfile`*`]`
ellipses ...	Ellipses indicate an arbitrary number of similar items: `CHKVAL fieldname value1 value2 ... valueN`
italics	Italic type indicates a variable. Substitute a value for the variable: *`library_name`*
vertical line |	A vertical line indicates a choice within braces or brackets: `FILE` *`filesize`* `[K|M]`
forward slash /	A forward slash is used as an escape character in the Oracle VM Manager Command Line Interface to escape the special characters `"`, `'`, `?`, `\`, `/`, `<`, `>`. Special characters need only be escaped when within single or double quotes: `create Tag name=MyTag description="HR/'s VMs"`

4 Conventions

The following text conventions are used in this document:

Convention	**Meaning**
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1 What's New in the Oracle VM Manager Command Line Interface (CLI)?

This section lists the syntax changes in the Oracle VM Manager Command Line Interface (CLI) from previous releases. Included in this section are any changes or deletions to existing commands and options, as well as new commands or options. This section may be useful when migrating your CLI scripts from previous releases of the CLI.

1.1 CLI Changes in Release 3.3

This section lists the changes and new features to the CLI in Release 3.3. In this release, a lot of changes have been introduced due to the move to the Web Services API.

Important

A major change to the way in which the CLI interacts with Oracle VM Manager means that CLI-based transactions have changed in Release 3.3. From Release 3.3 onwards, if a CLI command that contains child operations fails, it is important to check the state of all objects affected by the parent and child operations. For example, during the creation of a server pool, the parent command calls a number of child operations. If any of these child operations fail, the command returns an error, but the server pool object may still be created by the parent operation. In this case, the server pool object must be removed manually before attempting to create the server pool again.

1.1.1 CLI Modifications

This section lists the modifications made to existing CLI commands.

Table 1.1 CLI Modifications in Release 3.3

Object Type	Description
	The getEvents command has a completely different syntax.
AccessGroup	The edit AccessGroup command now includes a `nameOnArray` option to change the name on the storage array. **Note** This change was introduced after the initial release of Oracle VM 3.3, so may not be available if you are not running the latest patched version of this release.
AccessGroup (was NfsAccessGroup)	The `NfsAccessGroup` object type has been renamed and is now referred to as `AccessGroup`. This affects the following commands: • `create NFSAccessGroup` is now create AccessGroup • `edit NFSAccessGroup` is now edit AccessGroup • remove FileSystem option renamed • add FileSystem option renamed • add Server option renamed

Object Type	Description
	• remove Server option renamed • add ServerPool option renamed • remove ServerPool option renamed • delete option renamed • show option renamed • list option renamed
Assembly	The `createTemplatesFromAssembly` command has been removed and replaced with the new createVmFromAssembly command.
Assembly, VirtualCdrom, VirtualDisk, Vm	The following commands no longer require or have the `server` option to enter the Oracle VM Server to perform the import job: • importAssembly • importTemplate • importVirtualCdrom • importVirtualDisk • importVirtualMachine
BondPort, Port, VlanInterface	The `netmask` option for various commands has been renamed and is now referred to as `ipNetmask`. This affects the following commands: • add VlanInterface option renamed • create BondPort option renamed • edit BondPort option renamed • edit Port option renamed
BondPort, Port	The commands to manage ports and bond ports have changed. The following commands have been significantly modified: • create BondPort • edit BondPort • edit Port
FileServer	The create FileServer and edit FileServer commands no longer have the `adminHost`, `adminUsername`, and `adminPassword` options.
FileServer	The `addNfsRefreshServer` command has been renamed to addRefreshServer, and the `nfsRefreshServer` option for this command has been renamed to `server`. The `removeNfsRefreshServer` command has been renamed to removeRefreshServer, and the `nfsRefreshServer` option for this command has been renamed to `server`.

Object Type	**Description**
Job	The `getJobsUsingRange` and `getLatestNumberOfJobs` commands have been removed and are no longer available. Instead, use the getJobs command.
Network	The create Network option to create a local network on an Oracle VM Server, `server=`*`value`*, has been changed to `Server` *`instance`*.
PhysicalDisk	The `thinProvision` option of the create PhysicalDisk command is now mandatory.
PhysicalDisk, VirtualDisk	The edit PhysicalDisk and edit VirtualDisk commands no longer have the `size` option to enable resizing of a disk. Instead, there is a new resize command to perform this task for both virtual and physical disks.
Repository	The create Repository command has significantly changed to allow a repository to be created only on a file system and no longer allows you to create a repository on a physical disk.
Repository	The edit Repository command has removed the `ownership` option to take or release ownership of a repository. To perform this action, two new commands have been added, takeOwnership and releaseOwnership.
Server	The discoverServer command `username` option has been removed, the default username of `oracle` is used. The `port` option has been removed, the default port of `8899` is used.
Server	The add Server command options `FileServer` and `StorageArray` have been removed. These options set an Oracle VM Server as an admin server for those objects. Instead, you should now use the addAdminServer command. The remove Server command options `FileServer` and `StorageArray` have been removed. These options remove an Oracle VM Server as an admin server from those objects. Instead, you should now use the removeAdminServer command.
Server	The `takeOwnership` option has been removed from the edit Server command. Instead use the takeOwnership command to take ownership of an Oracle VM Server or the releaseOwnership command to release ownership. The `maintenanceMode` option has been removed from the edit Server command. Instead use the setMaintenanceMode command to set whether an Oracle VM Server is in maintenance mode.
Server	The `setNtp`, `showNtp` and `syncNtp` commands have been removed. Instead, use the edit Server command with the new `ntpServers` option to set or edit the time using one or more NTP servers on an Oracle VM Server. The show command with the `Server` option displays the NTP servers associated with an Oracle VM Server. There is no command that replaces the `syncNtp` command.
Server	The `initiateYumUpgrade` command has been removed. Instead, use the upgrade command to update the software on an Oracle VM Server.
ServerPool	The create ServerPool and edit ServerPool commands `keymapName` option now has a list of options available instead of the option having to be entered manually.
StorageArray (was SanServer)	The `SanServer` object type has been renamed and is now referred to as `StorageArray`. This affects the following commands:

Object Type	Description
	• `create SanServer` is now create StorageArray • `edit SanServer` is now edit StorageArray • `delete SanServer` is now delete `StorageArray` Other commands that relate to this object type, such as commands specific to `AccessGroup` and `Server` objects are also affected.
StorageArray	The create StorageArray and edit StorageArray commands have different values for the `storageType` option parameters.
VlanInterface (was VlanGroup and VlanSegment)	The commands to manage VLANs have changed. You should now use the VlanInterface object type instead of the VlanGroup and VlanSegment object types. The following commands have been removed: • `create VlanGroup` • `edit VlanGroup` • `create VlanSegment` • `edit VlanSegment` The following commands have been modified to remove the `VlanGroup` option: • add BondPort • remove BondPort • add Port • remove Port • list • show • delete The following commands have been modified to remove the `VlanSegment` option: • create Network • edit Network • list • show • delete The edit VlanInterface command no longer has the options `addressType`, `ipAddress`, or `ipNetmask`.

Object Type	Description
Vm	The add Vm and remove Vm commands have additional syntax options and functions. In addition to adding or removing a virtual machine to/from an Oracle VM Server, you can now add or remove a virtual machine to/from a server pool or an anti affinity group.
Vm	The clone Vm command `destName` option is now optional.
VmDiskMapping	The create VmDiskMapping command has changed. Instead of specifying the `storageDevice` option to map a disk, you must now specify either a `physcialDisk`, `virtualCd`, or a `virtualDisk`.
Vnic	You can no longer add a VNIC to a virtual machine. To support this change, the add Vnic command no longer allows to add a VNIC to a virtual machine. Instead, you may add a VNIC to either a `LocalNetwork` or `Network` object type. Similarly, the remove Vnic command now has a `LocalNetwork` or `Network` object type. In addition, the `addAvailableVnic Vm` command has been removed.
Vnic	The create Vnic command has a completely different syntax. The command now uses syntax that forces the specification of the virtual machine to which the VNIC must be attached. The edit Vnic command no longer allows you to specify the network (the `network` option as been removed), but does allow you to specify the VNIC MAC address with the new `macAddress` option.
Vnic	The `vnicCreate` command to create multiple VNICs has been removed as VNICs can no longer exist within Oracle VM Manager outside of the context of a virtual machine.
Vnic	The show command can now take either the ID or name of a VNIC to display the properties of a VNIC object. Previously, you could only use the VNIC name.
Vm	The `getIncompatibleReasons` command is no longer available.
Vm	The create Vm command syntax to create a virtual machine on an Oracle VM Server has changed. A new `server` option is available and the `on Server` option has been removed.
YumConfig	The `edit yumConfig` command is no longer available. Instead, use the create ServerUpdateRepository command to set a YUM repository for Oracle VM Server software updates. The `show YumConfig` command has also been removed.

1.1.2 CLI New Features

This section lists the new commands introduced in Release 3.3.

Table 1.2 CLI New Features in Release 3.3

Object Type	Description
N/A	The set command now includes options to set the `CommandMode` to be synchronous or asynchronous, and the `CommandTimeout` to set when the CLI times out.
N/A	The setStatsConfig and getStatsConfig commands set and display the statistics information displayed in Oracle VM Manager.
N/A	The getDebugTranscript command displays the debug transcript of a job.
All	The getEventsForObject command displays the events for an object.

Object Type	Description
All	The getTriageEvent command displays the highest severity event for an object.
AccessGroup	The refresh command has a new `AccessGroup` option to refresh the configuration information for an AccessGroup object.
AntiAffinityGroup	The create AntiAffinityGroup command provides a way to create an anti affinity group for virtual machines in a server pool. There is also an edit AntiAffinityGroup to edit an anti affinity group. The add Vm and remove Vm commands now allow you to add and remove virtual machines to and from an anti affinity group respectively using the `AntiAffinityGroup` option. The show, list, and delete commands include the `AntiAffinityGroup` option to support this new feature.
Assembly	A new createVmFromAssembly command replaces the now removed `createTemplatesFromAssembly` command.
Assembly	A new getDescriptor command lists the contents of the OVF descriptor file for the assembly.
AssemblyVirtualDisk	A new command edit AssemblyVirtualDisk allows you to edit an AssemblyVirtualDisk object type. The show, and list, commands now have the `AssemblyVirtualDisk` option to support this new feature.
AssemblyVm	A new command edit AssemblyVm allows you to edit an AssemblyVm object type. The show, and list, commands now have the `AssemblyVm` option to support this new feature.
Assembly, VirtualCdrom, VirtualDisk, Vm	The following commands include a `proxy` option to include a proxy server during the import job if required. • importAssembly • importTemplate • importVirtualCdrom • importVirtualDisk • importVirtualMachine
Bond, BondPort, VlanInterface	Three new commands allow editing of the ipAddressConfig object, which is an object associated with a parent network-related object that cannot be accessed directly with other CLI commands. This allows you to configure IP addresses on a port, bond port or VLAN interface. The new commands are: • embeddedCreate • embeddedEdit • embeddedDelete
ControlDomain	A new edit ControlDomain command allows you to edit the ControlDomain object type.

Object Type	Description
	The show, and list, commands now have a `ControlDomain` option to support this new feature.
Cpu	A new edit Cpu command allows you to edit the Cpu object type. The show, and list, commands now an have a `Cpu` option to support this new feature.
CPUCompatibilityGroup	New commands to access the `CPUCompatibilityGroup` object type: • create CpuCompatibilityGroup • edit CpuCompatibilityGroup Commands that contain a new `CPUCompatibilityGroup` option: • list • show • delete • add Server • remove Server
FileServer	The create FileServer command includes two new options to set `adminServers` and `refreshServers`.
FileServer	The edit FileServer command includes a new `plugin` option to set the Oracle VM Storage Connect plug-in for the file server.
FileServer, StorageArray	The addAdminServer command allows you to add an administrative Oracle VM Server to a file server or storage array. There is also a new removeAdminServer command that allows you to remove an administrative Oracle VM Server from a file server or storage array.
FileServer, FileSystem	The create FileSystem command creates an OCFS2 file system on a physical disk on a file server.
FileServerPlugin	A new edit FileServerPlugin command allows you to edit the FileServerPlugin object type. The show, and list, commands now an have a `FileServerPlugin` option to support this new feature.
Job	A new getJobs command lists jobs. A new getQueuedJobInfo command lists information about a queued job.
Manager	A new edit Manager command allows you to edit the Manager object type. The show, and list, commands now an have a `Manager` option to support this new feature.
PeriodicTask	A new edit PeriodicTask command allows you to edit the PeriodicTask object type. The show, and list, commands now an have a `PeriodicTask` option to support this new feature.

Object Type	Description
PhysicalDisk	The create PhysicalDisk command has a new mandatory `userFriendlyName` option.
PhysicalDisk, VirtualDisk	The resize command resizes a physical or virtual disk.
Repository	The show command with the `Repository` option displays a new parameter to show the share path.
RepositoryExport	The create RepositoryExport command provides a way to create a repository export to enable back up of an OCFS2-based storage repository. The show, list, and delete commands now have a `RepositoryExport` option to support this new feature. The `show Server` command now lists the repository exports on an Oracle VM Server.
Server	The setMaintenanceMode command sets whether an Oracle VM Server is in maintenance mode.
Server	The create ServerUpdateGroup command creates an object to contain a repository for updating the software on Oracle VM Servers. This group can be associated with a single server pool. There is also an associated edit ServerUpdateGroup command. The show, list, and delete commands include the `ServerUpdateGroup` option to support this new feature.
Server	The create ServerUpdateRepository command sets a repository for updating the software on Oracle VM Servers. This repository is then used in a server update group. There is also an associated edit ServerUpdateRepository command. The show, list, and delete commands include the `ServerUpdateRepository` option to support this new feature.
Server	The checkUpToDate command checks whether the Oracle VM Server software is up-to-date, according to the server update repository.
Server	The edit Server command now includes the `ntpServers` option to set one or more NTP servers on an Oracle VM Server to set the time. It also includes a `runVmsEnabled` option to set whether to allow additional virtual machines to run on the server.
Server	The show command with the `Server` option displays several new parameters: • The new `Up To Date` parameter displays whether a software update is available for an Oracle VM Server from a server update repository. • The new `Control Domains` parameter displays the details of any control domains that are running on the Oracle VM Server instance. • The new `CPU Compatibility Groups` parameter describes the *server processor compatibility group* to which the Oracle VM Server belongs.
Server	The changeServerAgentPassword allows you to change the Oracle VM Agent password on an Oracle VM Server.
Server, FileServer, StorageArray	The refreshAll command rediscovers all Oracle VM Server instances, file servers, and storage arrays.

Object Type	**Description**
Server, Repository	A new takeOwnership command provides a way to take ownership of an Oracle VM Server or storage repository. There is also an associated releaseOwnership command.
ServerController	A new create ServerController command provides a way to configure IMPI on an Oracle VM Server. There is also an associated edit ServerController command. The show, list, and delete commands include the `ServerController` option to support this new feature.
ServerPool	The create ServerPool and edit ServerPool commands have new options for setting the cluster timeout (`clusterEnable`, and `clusterTimeout`) and setting the server pool policy (`policyMode`, `policyCpuEnable`, `policyPeriod`, and `policyCpuThreshold`).
ServerPool	The add ServerPool command has a new `AccessGroup` option. This option allows you to add a server pool to a file server access group (previously called NFS access group). In addition, the remove ServerPool command has a new `AccessGroup` option. This option allows you to remove a server pool from a file server access group.
ServerPoolNetworkPolicy	The create ServerPoolNetworkPolicy and edit ServerPoolNetworkPolicy commands provide a way manage the network policy for a server pool. The addPolicyServer and removePolicyServer commands allow to you add a policy server to a server pool, or to remove one. The show, list, and delete commands now have a `ServerPoolNetworkPolicy` option to support this new feature.
StorageArray	The create StorageArray command includes a `pluginPrivateData` option to allow the specification of additional Oracle VM Storage Connect plug-in parameters that are outside the scope of the parameters used to attach a generic storage array.
StorageArray	The create StorageArray and edit StorageArray commands have a new `storageName` option.
StorageArray	The edit StorageArray command includes a new `plugin` option to set the Oracle VM Storage Connect plug-in for the storage array.
StorageArray	The validate command validates a storage array using the storage array plug-in. Validation is required after the storage array is discovered and after modification of storage array attributes.
StorageArrayPlugin	The edit StorageArrayPlugin command allows you to edit the StorageArrayPlugin object type. The show, and list, commands now an have a `StorageArrayPlugin` option to support this new feature.
VirtualCdrom	The edit VirtualCdrom command now includes a `sharable` option to make the virtual CDROM/ISO file shareable.
VlanInterface	The commands to manage VLANs have changed significantly. The following commands have been introduced: • create VlanInterface

Object Type	Description
	• add VlanInterface • remove VlanInterface The following commands now include a `VlanInterface` option: • delete • list • show
Vm	The create Vm and edit Vm commands have an new `startPolicy` option to set the virtual machine start up policy. The `show Vm` command also shows the value for this parameter if it is set. The create Vm and edit Vm commands have an new `hugePagesEnabled` option to set whether to use HugePages.
Vm	The add Vm and remove Vm have a `ServerPool` option to add or remove a virtual machine to or from a server pool.
Vm	New commands are available to list all, or clear one or all messages sent or received by a virtual machine: • getVmReceivedMessages • getVmSentMessages • clearVmSentMessage • clearVmRcvdMessage • clearVmAllSentMessages • clearVmAllRcvdMessages
Vm	The getVmOsTypes command displays the operating system types used when creating or editing a virtual machine.
Vm	The getVmCfgFileContent command displays the contents of the virtual machine configuration file.
Vm, Repository	The moveVmToRepository command allows you to move a virtual machine to another storage repository.
VmCloneCustomizer, VmCloneNetworkMapping, VmCloneStorageMapping	The ability to add and manage virtual machine clone customizers is now available in the CLI. This includes the introduction of three new object types: • `VmCloneCustomizer` • `VmCloneNetworkMapping` • `VmCloneStorageMapping` As with most object types, you can use the respective `create`, `edit`, `delete`, `show`, and `list` commands to manage these new object types: • create VmCloneCustomizer

Object Type	Description
	• edit VmCloneCustomizer • create VmCloneNetworkMapping • edit VmCloneNetworkMapping • create VmCloneStorageMapping • edit VmCloneStorageMapping • list • show • delete This feature also facilitates a variety of cloning options. Newly added cloning commands are: • cloneCdToRepo • clonePdToPd • clonePdToRepo • clonePdToStorageArray • cloneVdToPd • cloneVdToRepo • moveVmToRepository The clone Vm command also has a new `cloneCustomizer` option to include a clone customizer, and a `targetRepository` option to specify the target repository.
Vnic	The setVnicMacAddrRange command changes the range of MAC addresses that are available to use for newly created VNICs. The getVnicMacAddrRange lists the MAC address range.

1.1.3 CLI Object Attribute Changes

This section lists the changes to the attributes of CLI objects in Release 3.3.1. The attribute changes affect the output of the `show` command, and the options available with `create` *`object`* and `edit` *`object`* commands.

Table 1.3 CLI Object Attributes Changes in Release 3.3.1

Object	Attributes Added	Attributes Removed	Attributes Modified
AccessGroup	agType storageArray fileSystems servers	sanServer	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.

Object	Attributes Added	Attributes Removed	Attributes Modified
	fileServer locked		
Assembly	origin locales assemblyVms assemblyVirtualDisks locked		Attribute repository's display name has changed. Old display name is `Repository Id`. New display name is `Repository`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
BondPort	ipAddressConfig network portType vlanInterfaces locked	ethernetPort1 ethernetPort2 addressType ipAddress netMask	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
FileServer	fstype storageDescription refreshServers server locked	nfsRefreshServers adminHost adminUserName adminPassword	Attribute uniformedExports's display name has changed. Old display name is `UniformedExports`. New display name is `UniformExports`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
FileSystem	clusterId fileServers accessGroup physicalDisk shared locked	fileServer	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Job	runState	uniqueId	

Object	Attributes Added	Attributes Removed	Attributes Modified
	summaryState done summaryDone jobGroup childJobs creationTime creationTimeLong startTimeLong endTimeLong durationDouble progressMessage abortedUser isAborted isAbortable id name locked	timestamp status	
ManagedObj	locked		Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Network	isLocal mtu roles vlanInterfaces locked	vlanSegment	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
PhysicalDisk	deviceNames storageTargetNames page83Id qos	sanserver	Attribute volumeGroup's display name has changed. Old display name is `VolumeGroup`. New display name is `Volume Group`.

Object	Attributes Added	Attributes Removed	Attributes Modified
	serverReserved state status pdtype userFriendlyName vendor accessGroups fileSystems reservingServers vmDiskMappings locked		Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Port	ipAddressConfig network bondPort portType vlanInterfaces locked	addressType ipAddress netMask	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Repository	managerUuid repoPresented virtualCdroms locked	serverPool physicalDisk ownership VirtualCdroms templates	Attribute fileSystem's display name has changed. Old display name is `FileSystem`. New display name is `File System`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Server	roles runVmsEnabled agentLogin statisticInterval ntpServers	MACAddress numProcessors numISCSIPorts numEthernetPorts numFibrePorts	Attribute ipAddress's display name has changed. Old display name is `IP Address`. New display name is `Ip Address`. Attribute memory's display name has

Object	Attributes Added	Attributes Removed	Attributes Modified
	biosVendor biosVersion biosReleaseDate processorType populatedProcessorSockets threadsPerCore coresPerProcessorSocket totalProcessorCores enabledProcessorCores usableMemory noExecuteFlag ovmVersion hostName managerUuid serverAbilities productSerialNumber upToDate controlDomains cpus controllers repositoryExports networks fileServerPlugins storageArrayPlugins storageArray storageArrayPlugin fileServerPlugin refreshFileServers AccessGroups locked	memoryOverhead networkFailOverGroupsCount architectureType serialNumber processorFamily processorModel takeOwnership l1CacheSize l2CacheSize l3CacheSize BIOSVersion BIOSReleaseDate socketsAvailable socketsFilled managerUUID version	changed. Old display name is `Memory (GiB)`. New display name is `Memory (MB)`. Attribute cpuCompatibilityGroup's display name has changed. Old display name is `CPU Compatibility Group`. New display name is `Cpu Compatibility Group`. Attribute tags' display name has changed. Old display name is `tag`. New display name is `Tag`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.

Object	**Attributes Added**	**Attributes Removed**	**Attributes Modified**
ServerPool	antiAffinityGroups clusterTimeout policyId clusterId heartbeatDeviceId managerUuid poolFileSystem policyMode policyCpuEnable policyPeriod policyCpuThreshold policyServers networkPolicies serverUpdateGroup locked		Attribute tags' display name has changed. Old display name is `tag`. New display name is `Tag`. Attribute startPolicy's display name has changed. Old display name is `startPolicy`. New display name is `StartPolicy`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
StorageInitiator	accessGroupIds initType fcFabricName fcPortId fcPortType fcPortSpeed fcSupportedSpeeds fcNodeName locked		Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Tag	locked		Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
VirtualCdrom	vmDiskMapping shareable	vms	Attribute description's display name has

Object	Attributes Added	Attributes Removed	Attributes Modified
	assemblyVirtualDisk importFileName locked		changed. Old display name is `description`. New display name is `Description`.
VirtualDisk	file mountPoint vmDiskMapping assemblyVirtualDisk importFileName locked	status vms	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
VlanInterface	interfaceName port portType vlanId ipAddressConfig network macAddress locked	vlansegment addressType ipAddress netMask	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Vm	currentMemory currentCpuCount startPolicy bootOrder diskLimit hugePagesEnabled serverPool vmCloneCustomizers locked	cpuCompatibilityGroupName networkBootPath	Attribute tags' display name has changed. Old display name is `tag`. New display name is `Tag`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
VmDiskMapping	slot virtualDisk virtualCd physicalDisk	storageDevice	Attribute vm's display name has changed. Old display name is `Vm Id`. New display name is `Vm`.

Object	Attributes Added	Attributes Removed	Attributes Modified
	locked		Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
Vnic	ipAddresses interfaceName locked	ipAddress	Attribute vm's display name has changed. Old display name is `Vm Id`. New display name is `Vm`. Attribute network's display name has changed. Old display name is `Network Id`. New display name is `Network`. Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.
VolumeGroup	allocatedSize volumeDescription storageArray locked	sanserver	Attribute description's display name has changed. Old display name is `description`. New display name is `Description`.

Chapter 2 Introduction to the Oracle VM Manager Command Line Interface (CLI)

The Oracle VM Manager Command Line Interface (CLI) provides a command line interface to communicate with Oracle VM Manager. You can use the CLI to perform many of the same functions as Oracle VM Manager, such as managing all your server pools and *guest* virtual machines. The CLI commands can be scripted to enable flexibility to help you deploy and manage your Oracle VM environment.

The CLI is installed when you install Oracle VM Manager, so you must have a working copy of Oracle VM Manager to use the CLI. The architecture of the CLI is shown in Oracle VM Architecture in the *Oracle VM Concepts Guide*.

When you make changes to the Oracle VM environment using the CLI, these changes are reflected in real time in the Oracle VM Manager Web Interface.

The Oracle VM Manager Web Interface includes additional logic over the CLI to make sure that actions performed within Oracle VM Manager do not result in configurations that may cause runtime errors. This additional logic is not available within the CLI, which provides greater flexibility, but requires a deeper understanding of Oracle VM object relationships.

The CLI does not replace the Oracle VM Utilities. The Oracle VM Utilities are complementary to the CLI. For information on the Oracle VM Utilities, see the *Oracle VM Administrator's Guide*.

2.1 Starting and Stopping the CLI

The CLI is automatically started when you first install Oracle VM Manager. You can also stop and start it separately to Oracle VM Manager as required. To use the CLI, the Oracle VM Manager service should first be started. See Starting and Stopping Oracle VM Manager for information on starting and stopping Oracle VM Manager.

When the CLI is installed (as part of Oracle VM Manager), it is set to automatically start when the operating system starts. If you want to disable this, enter:

```
# chkconfig ovmcli off
```

To manually start or stop the CLI, and to verify its status, log in as the *root* user, and use the syntax:

```
/sbin/service ovmcli [start|stop|status|restart]
```

For example, to start the CLI:

```
# /sbin/service ovmcli start
```

To stop the CLI:

```
# /sbin/service ovmcli stop
```

To restart the CLI:

```
# /sbin/service ovmcli restart
```

To check the status of the CLI:

```
# /sbin/service ovmcli status
```

The `status` option returns whether the CLI service is running or stopped.

If the Oracle VM Manager *host computer* runs a full graphical desktop environment, you can also use the **Services** dialog to start and stop the CLI. From the **Applications** menu, select **System Settings** > **Server Settings** > **Services**. Or by running the following command in a terminal to display the **Services** dialog:

```
# /usr/bin/system-config-services
```

In the **Service Configuration** dialog, select **ovmcli** to check the status, and start or stop it.

2.2 Connecting to the CLI

Multiple CLI connections can be made to a single instance of Oracle VM Manager at any time. The connection to the CLI is an SSH connection. To connect to the CLI, use an SSH client or command line interface and connect to the Oracle VM Manager host using the syntax:

`ssh -l` *`manager_username`* { *`manager_IP`* | *`manager_hostname`* } `-p` *`port`*

The default port for the CLI is 10000.

For example:

```
$ ssh -l admin 10.172.76.146 -p 10000
```

To connect to the CLI from the Oracle VM Manager host, enter:

```
$ ssh -l admin localhost -p 10000
```

You can also use the abbreviated connection syntax (without the `-l` option), for example:

```
$ ssh admin@localhost -p 10000
```

Tip

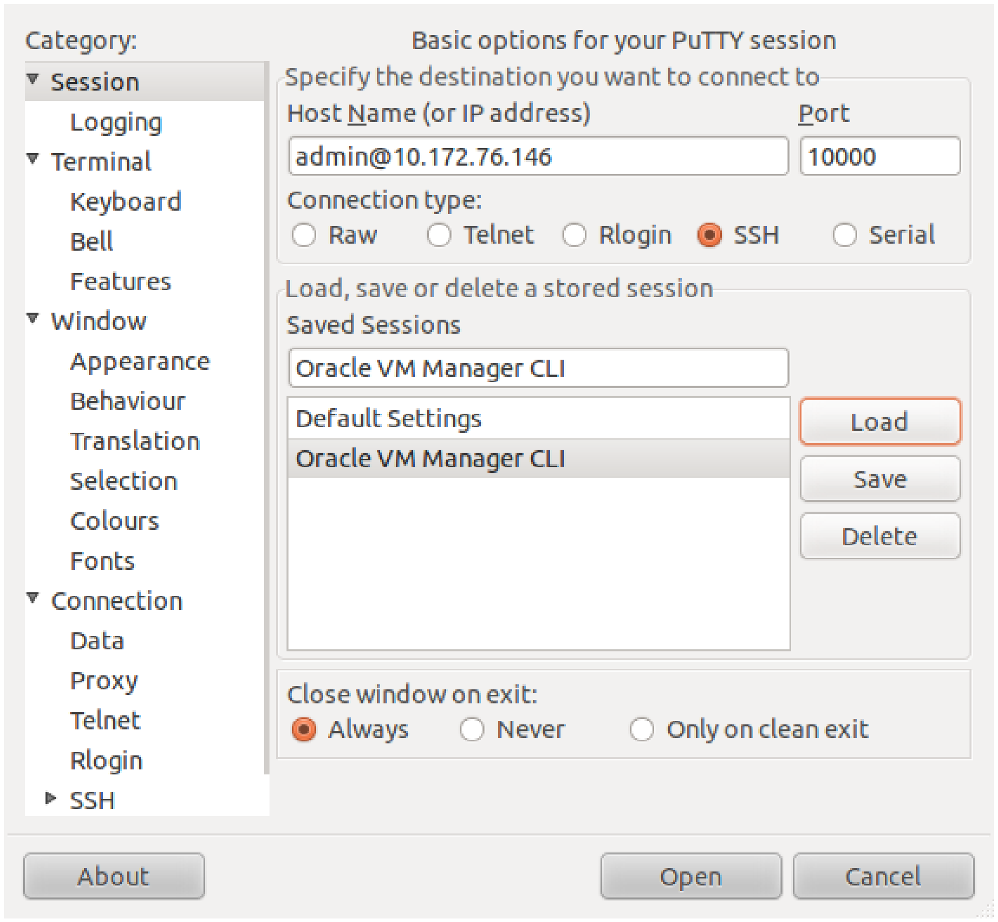
To keep your ssh session from disconnecting due to inactivity, you can use the ssh ServerAliveInterval option to send a null packet to the CLI to keep the connection alive. You can use this option, either on the command line when you enter the ssh command, or in the ~/.ssh/config file, for example to use this on the command line, enter:

```
$ ssh admin@localhost -p 10000 -o ServerAliveInterval=40
```

The interval length is in seconds, so this example will keep the connection alive for 40 seconds. Setting this value to `0` disables the feature. The syntax to use when adding this to the ~/.ssh/config file is:

```
Host *manager_hostname
 ServerAliveInterval 40
```

Alternatively, you can use a graphical SSH client like PuTTY:



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): admin@10.172.76.146

Port: 10000

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: Oracle VM Manager CLI

Default Settings
Oracle VM Manager CLI

Close window on exit: [✓ Always] Never Only on clean exit

You are prompted to enter a password for the Oracle VM Manager admin user. Enter the password and the CLI prompt is displayed, ready for you to begin entering Oracle VM CLI commands.

```
Using username "admin".
admin@10.172.76.146's password: password
OVM>
```

To exit the CLI, enter `exit`, or end the SSH session.

2.3 SSH Host Keys

On some operating systems, when you first log in to the CLI, you may be prompted to add the key fingerprint of the Oracle VM Manager host to the `~/.ssh/known_hosts` file, for example:

```
$ ssh -l admin hostname -p 10000
The authenticity of host 'hostname (IP_address)' can't be established.
DSA key fingerprint is fingerprint.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hostname' (DSA) to the list of known hosts.
admin@hostname's password:
```

If you want to avoid this message and have host keys automatically added to the `known_hosts` file, you can turn off strict checking of SSH host keys using the following command:

```
$ ssh -o 'StrictHostKeyChecking no' admin@hostname
```

If you have upgraded or reinstalled Oracle VM Manager, you may be prompted that the host identification has changed when connecting to the CLI, for example:

```
$ ssh -l admin hostname -p 10000
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the DSA host key has just been changed.
The fingerprint for the DSA key sent by the remote host is
fingerprint.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending key in ~/.ssh/known_hosts:1
DSA host key for localhost has changed and you have requested strict checking.
Host key verification failed.
```

To clear this message and allow connections to the CLI, use the `ssh-keygen` utility to remove the entry for the Oracle VM Manager host, for example:

```
$ ssh-keygen -R hostname:10000
```

You may need to add braces around the hostname if the previous command does not remove the entry, for example:

```
$ ssh-keygen -R [localhost]:10000
```

Alternatively, edit the `~/.ssh/known_hosts` file and remove the entry for the Oracle VM Manager host.

Connect to the CLI again and you are prompted to add the new fingerprint if strict checking of SSH host keys is enabled.

2.4 Key-Based Authentication

You can use public key-based SSH authentication to connect to the CLI. When you set up key-based authentication, you can log in to the CLI without being prompted for a password. Using key-based authentication does not effect the existing authentication mechanism using a username and password.

Keys are set up between a local system and the Oracle VM Manager host. After the keys are set up, you log in for the first time with the CLI admin password, as you normally would, then a connection is established using public key authentication and the channel is identified to the CLI Server by the client IP address and user name. Subsequent log ins do not require the password, for as long as the channel remains open. If the channel is closed, or the admin user's password is changed, key-based authentication is terminated and you are required to enter the admin user's password again. To reestablish the connection using key-based authentication, log in again using the standard SSH connection, enter the admin user's password, and the channel is opened again.

To set up key-based authentication:

1. Make sure the ssh-agent is running on your local host:

   ```
   $ eval `ssh-agent`
   Agent pid number
   ```

   If the ssh-agent is not running, you may encounter the following error when you perform the next step:

   ```
   Could not open a connection to your authentication agent.
   ```

2. On your local host, generate a public/private key pair to log in to the CLI. Make sure you remember the passphrase that you enter.

   ```
   $ ssh-keygen -t rsa -f ~/.ssh/admin
   Generating public/private rsa key pair.
   Enter passphrase (empty for no passphrase): passphrase
   ```

```
Enter same passphrase again: passphrase
Your identification has been saved in /user/.ssh/admin.
Your public key has been saved in /user/.ssh/admin.pub.
The key fingerprint is:
fingerprintuser@hostname
```

The two keys are generated in `~/.ssh/`: `admin` (the private key) and `admin.pub` (the public key).

3. Add the private key to the authentication agent, using the same passphrase you used to create the key pair, for example:

   ```
   $ ssh-add ~/.ssh/admin
   Enter passphrase for /home/user/.ssh/admin: passphrase
   Identity added: /home/user/.ssh/admin (/home/user/.ssh/admin)
   ```

4. Copy the public key to the Oracle VM Manager host, for example:

   ```
   $ scp ~/.ssh/admin.pub oracle@hostname:/home/oracle/.ssh/
   ```

   Where *`hostname`* is the hostname of the v host. Make sure you do the copy as the oracle user.

5. Log into the Oracle VM Manager host as the oracle user and append the `admin.pub` public key to the CLI authorized file (`ovmcli_authorized_keys`).

   ```
   $ ssh oracle@hostname
   $ cd /home/oracle/.ssh/
   $ cat admin.pub >> ovmcli_authorized_keys
   $ exit
   ```

   Where *`hostname`* is the hostname of the Oracle VM Manager host.

6. From the local machine, log in to the CLI using the command:

   ```
   $ ssh -l admin hostname -p 10000
   ```

   You are requested to enter the admin user's password. Enter it.

   ```
   admin@hostname's password: password
   OVM>
   ```

   Subsequent log ins use the newly established channel and do not require a password.

For security reasons, the channel for public key authentication expires after a designated period of time. See Section 2.10, “Configuring the Oracle VM CLI” for information on how to change the public key authentication expiry time.

2.5 Using SSH to Run Background Processes

If you choose to run CLI commands in the background via the standard SSH interface, it is possible that keyboard breaks interrupt running processes. This is a standard behaviour if the TTY setting has been set to `sane`, and is remedied by providing the SSH command with the `-n` option to automatically send keyboard input to `/dev/null`. This type of command can only be used with key-based authentication, since at the point that it is run, the command is run in the background so you can no longer interact with it.

```
$ ssh -l admin localhost -p 10000 -n "importAssembly Repository name=MyRepository
url=http://example.com/myassembly.ova" &
```

Note that using the `-n` flag does not make sense if you intend to maintain an interactive session within the CLI.

2.6 Using the Oracle VM CLI and Getting Help

Enter `?` or `help` to display help on a token. You can also enter `?` after a token to display the possible options based on context. For example, if you want to display information about an Oracle VM Server, you can work your way through the command options to find the commands to perform this action.

Note

To keep the output to a minimum in the examples in this book, we have set the output mode to sparse using the following command:

```
OVM> set OutputMode=Sparse
```

Your output may vary depending on which setting you use for this command; see Section A.147, “set” for more information.

To find the command to list Oracle VM Servers, start with the `?` option and work your way through the commands:

```
OVM> ?
     add
     create
     delete
     edit
     embeddedcreate
     embeddeddelete
     embeddededit
     exit
     help
     list
     Perhaps this is the command? Let's drill down further.
     remove
     set
     show
     showallcustomcmds
     showcustomcmds
     showobjtypes
     showversion
OVM> list ?
          AccessGroup
          AntiAffinityGroup
          Assembly
          ...
          Port
          Repository
          RepositoryExport
          Server
     This looks like the command to use to list Oracle VM Servers
          ServerController
          ServerPool
          ServerPoolNetworkPolicy
          ...
          VmDiskMapping
          Vnic
          VolumeGroup
OVM> list Server
     No more options can be entered so the results are automatically displayed
  id:00:e0:81:4d:5f:2f:00:e0:81:4d:29:ee:00:e0:81:4d  name:MyServer1
  id:00:e0:81:4d:5e:16:00:e0:81:4d:5e:17:ff:ff:ff:ff  name:MyServer2
  id:00:e0:81:4d:40:16:00:e0:81:4d:40:17:ff:ff:ff:ff  name:MyServer3
OVM>
```

Now you have a list of the Oracle VM Servers, you can display information about them with another command. To find the command to display information about an Oracle VM Server, drill down again through the commands to find the most appropriate command using the `?` option:

```
OVM> ?
     add
     create
     delete
     edit
     embeddedcreate
     embeddeddelete
     embeddededit
     exit
     help
     list
     remove
     set
     show
       This looks like the command to use to show information
     showallcustomcmds           commands available for all objects
     showcustomcmds              commands specific to an object (requires object as argument)
     showobjtypes
     showversion
OVM> show ?
          AccessGroup
          AntiAffinityGroup
          Assembly
          ...
          Port
          Repository
          RepositoryExport
          Server
       This looks like the command to use to show information about an Oracle VM Server
          ServerController
          ServerPool
          ServerPoolNetworkPolicy
          ...
          VmDiskMapping
          Vnic
          VolumeGroup
OVM> show Server ?
                 id=<object identifier> OR
                 name=<object name>
OVM>
```

If you have forgotten the name of the Oracle VM Server, use the up arrow to scroll through the history until you see the `list Server` command and press **Enter**. Then use the `show Server name=` option to display information about an Oracle VM Server.

```
OVM> show Server name=MyServer1
  Status = Running
  Role 1 = Utility
  Role 2 = Vm
  Ip Address = 10.172.76.79
  ...
  Name = MyServer1
  Locked = false
OVM>
```

The CLI is a self-learning tool; built in help and tab auto-completion guide you when working with the commands. The following commands assist you to use the CLI.

Table 2.1 Helpful CLI commands

Command/Feature	Description
`?`	Context sensitive help, for example, `show ?`, `clone ?`. If you do not know the format of a command, enter the command followed by `?` to see the options for that command. Enter `?` on its own to see a list of all the top level commands.

Command/Feature	Description
`help`	Displays the syntax to use for the top level commands.
`showallcustomcmds`	Displays a list of the all custom commands for all object types.
`showcustomcmds [object type]`	Displays a list of the custom commands for a specific object type provided as a parameter.
`showobjtypes`	Displays a list of the object types.
tab completion	Press the **Tab** key to auto-complete the command.
history	Use the up or down arrow keys to step through the history of commands entered in the current session. Up to 50 commands are listed.

You can configure the end of line characters used by your SSH client, for example, if your SSH client adds a line feed (double spacing) to the end of a line, you can set the endline characters to `CR`. Set the end of line characters using the set command.

You can configure the output mode to define how the CLI returns results, for example in plain text or in XML. Set the output mode using the set command.

The values you enter for parameters are case sensitive, unless explicitly stated in this Guide. For example, entering `name=MyServer` is not the same as entering `name=myserver`. The CLI treats these parameter strings as case sensitive, and are considered different.

Special characters are considered any of these: `"`, `'`, `?`, `\`, `/`, `<`, `>`. You can escape special characters within a set of quotes to make sure they are treated as a literal string using a `/` (forward slash) before the character. For example:

```
OVM> create Tag name=MyTag description="HR/'s VMs from http:////example.com//vms// /<Delete/?/>"
  id:0004fb0000260000b351e52e3abbe192  name:MyTag
OVM> show Tag name=MyTag
  Name = MyTag
  Id = 0004fb0000260000b351e52e3abbe192
  description = HR's VMs from http://example.com/vms/ <Delete?>
```

Parameters are unique for each time you run a command. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence. Therefore, a command similar to the following succeeds, but the values of these repeated parameters override each other:

```
OVM> discoverServer ipAddress=server1 ipAddress=server2 takeOwnership=No\
 takeOwnership=Yes password=** password=******
 Command: discoverServer ipAddress=server1 ipAddress=server2 takeOwnership=No
 takeOwnership=Yes password=** password=******
 Status: Success
 Time: 2013-12-23 00:34:38,398 PST
 JobId: 1387787665552

OVM> list server
 Command: list server
 Status: Success
 Time: 2013-12-23 00:34:43,602 PST
 Data: id:44:45:4c:4c:59:00:10:35:80:34:c7:c0:4f:57:48:31
       name:server2
```

In this example, only one Oracle VM Server is discovered. The second parameter in each instance of the command overrides the first. Therefore, `server2` is discovered, and ownership is set to `Yes`, and the password in this instance is the second one specified.

If a command is issued, but no changes to an object are performed, a success message is displayed. For example, if you change the name of an object to the same name, `Status: Success` is returned and displayed.

2.6.1 Return Status Values

Operations that trigger job creation within Oracle VM Manager, such as create or modify operations, always return the status of the job. The status is referred to as the Return Status. If a job is successful, the status returned is set to **Success**. If a job fails, the status returned is set to **Failure**.

Job status is obtained by consistently polling Oracle VM Manager with the generated Job ID to query the job status. For operations that may take too long, the CLI may timeout the polling period. In this case, the status returned is set to **Running** to indicate that the job is still running. In this case, you may need to query the job status manually before continuing with any other operations.

Since all commands that result in some form of change within Oracle VM Manager trigger the creation of a job, these commands also return the job ID in the response output. The value for this appears in the **JobId** field. In the case where a command has returned a **Running** status, you can use this field to obtain the job ID for the task that you are performing, and then use the `show job` command to monitor its status.

The default job timeout value is 7200 seconds (120 minutes). The value can be set in the common configuration file, used by both the Oracle VM Manager CLI and the Oracle VM Manager Web Interface, at `/u01/app/oracle/ovm-manager-3/ovm_cli/config/common_config.xml`. The attribute to set is called **defaultCommandTimeout**.

2.7 Passing in a CLI Command at the Command Line

If you want to run the CLI and pass in a simple command, append the command in quotes after the SSH login credentials, for example:

```
# ssh admin@hostname -p 10000 "list Server"
```

You can submit multiple commands using a semicolon (;) as the command delimiter, for example:

```
# ssh admin@hostname -p 10000 "list Server; show Server name=MyServer"
```

If you have enabled key-based authentication, you are not prompted for the password to access the CLI, and the results are displayed. If you have not, you must enter the CLI password before the results are returned. See Section 2.4, “Key-Based Authentication” for information on setting up key-based authentication. You can also use any other programming language to write scripts using the CLI as discussed in Section 2.9, “Integrating the CLI into Your Applications”.

2.8 Sample Scripts

There are a number of shell and Expect scripts provided with Oracle VM to help you use the CLI. These scripts are located in:

`/u01/app/oracle/ovm-manager-3/ovm_cli/expectscripts`

Warning

Some of the sample scripts store the Oracle VM Manager login credentials in plain text. These scripts should not be used in a production environment, and especially should not be used on a machine with low security settings. You should implement your own security methods for the login credentials.

Table 2.2 Sample scripts

Script Name	Purpose

`ackAllEvents`	An Expect script that acknowledges all events that a user can acknowledge. This script does not acknowledge system events.
	
	To use this script, edit it and change the following lines to include the login credentials for Oracle VM Manager:
	
	`set username username`
	`set password password`
	
	To run the script, enter:
	
	`# ./ackAllEvents`

`inventory`	An Expect script that displays an inventory of all objects managed by Oracle VM Manager.
	
	To use this script, edit it and change the following lines to include the login credentials for Oracle VM Manager:
	
	`set username username`
	`set password password`
	
	To run the script, enter:
	
	`# ./inventory`

There are a also a set of Expect scripts provided to get you started scripting in the CLI. These scripts are located in:

`/u01/app/oracle/ovm-manager-3/ovm_cli/expectscripts/createdeletescripts`

A text file named `README` in the directory provides information on using these scripts and may have additional information not included in this section.

Warning

Some of the sample scripts store the Oracle VM Manager login credentials in plain text. These scripts should not be used in a production environment, and especially should not be used on a machine with low security settings. You should implement your own security methods for the login credentials.

Table 2.3 Sample create/delete scripts

Script Name	Purpose

`create-fc-based-VM.cli`	An Expect script that sets up a complete Oracle VM environment, including storage, server pool, networking, virtual machine resources and a virtual machine, using fibre channel-based storage.
	
	To use this script, edit the `fc-based-VM.properties` file in the same directory and change the parameters to suit your environment. To run the script, enter:
	
	`# ./runOVMCLITest  -test=create-fc-based-VM.cli \`
	`    -arguments=fc-based-VM.properties`

`delete-fc-based-VM.cli`	An Expect script that deletes the set up created using the `create-fc-based-VM.cli` script.

Script Name	Purpose

	To use this script, edit the `fc-based-VM.properties` file in the same directory and change the parameters to suit your environment. To run the script, enter:
	`# ./runOVMCLITest  -test=delete-fc-based-VM.cli \`
	`    -arguments=fc-based-VM.properties`

`create-iscsi-based-VM.cli`	An Expect script that sets up a complete Oracle VM environment, including storage, server pool, networking, virtual machine resources and a virtual machine, using ISCSI-based storage.
	
	This script uses a fibre channel LUN for the server pool's clustered file system. The Oracle VM Server must have access to an unmanaged fibre channel storage array. This script creates a storage array using the Oracle NetApp Filer plug-in.
	
	To use this script, edit the `iscsi-based-VM.properties` file in the same directory and change the parameters to suit your environment. To run the script, enter:
	`# ./runOVMCLITest  -test=create-iscsi-based-VM.cli \`
	`    -arguments=iscsi-based-VM.properties`

`delete-iscsi-based-VM.cli`	An Expect script that deletes the set up created using the `create-iscsi-based-VM.cli` script.
	
	To use this script, edit the `iscsi-based-VM.properties` file in the same directory and change the parameters to suit your environment. To run the script, enter:
	`# ./runOVMCLITest  -test=delete-iscsi-based-VM.cli \`
	`    -arguments=iscsi-based-VM.properties`

`create-nfs-based-VM.cli`	An Expect script that sets up a complete Oracle VM environment, including storage, server pool, networking, virtual machine resources and a virtual machine, using NFS-based storage.
	
	To use this script, edit the `nfs-based-VM.properties` file in the same directory and change the parameters to suit your environment. To run the script, enter:
	`# ./runOVMCLITest  -test=create-nfs-based-VM.cli \`
	`    -arguments=nfs-based-VM.properties`

`delete-nfs-based-VM.cli`	An Expect script that deletes the set up created using the `create-nfs-based-VM.cli` script.
	
	To use this script, edit the `nfs-based-VM.properties` file in the same directory and change the parameters to suit your environment. To run the script, enter:
	`# ./runOVMCLITest  -test=delete-nfs-based-VM.cli \`
	`    -arguments=nfs-based-VM.properties`

`runAllCreateDeleteTests.sh`	An Expect script that runs all the create/delete scripts provided in the directory.

Script Name	Purpose
	To use this script, edit the `*-based-VM.properties` files in the same directory and change the parameters to suit your environment. To run the script, enter: `# ./runAllCreateDeleteTests.sh`
`runOVMCLITest`	A shell script to run the `*-based-VM.cli` Expect scripts which create/delete the Oracle VM environment. To run the script, enter: `# ./runOVMCLITest  -test=cli_script -arguments=properties_file`
`commonExpectDef.cli`	An Expect script that contains the common functions used by other scripts in this directory. Do not run this script directly.
`runCLI.py`	A Python script used by runOVMCLITest that reads the arguments defined in the property files and generates Expect scripts. Do not run this script directly.

To customize any of these scripts for your environment, copy them to a directory outside of the Oracle VM Manager installation tree, edit the script with a text editor and change the variables to suit your environment, or extend further to provide additional functionality.

When you run an Expect script, the output is displayed to the screen; spool it to a file or other process or script as required.

2.9 Integrating the CLI into Your Applications

CLI-based integration is a popular and inexpensive way to integrate Oracle VM into your own applications. It is far easier and faster than using the Web Services API, and for smaller integrations is a very good option.

Since it is possible to configure the CLI to return information in XML format, it is easy to develop applications that are capable of parsing the results of any command issued to the CLI. Set the output mode using the set command.

The connection information is cached on the Oracle VM Manager host for 15 minutes, so subsequent calls with the same login credentials are faster than the initial connection.

You can write your own CLI scripts and take advantage of all the rich constructs that the operating system shell provides: variables, looping, conditional execution, parsing, and so on.

An easy way to integrate with Java is to use the `exec` method within the `Runtime` class, then parse the XML output with an XML parser such as SAX.

Equally, scripts may be written using Python along with an XML parser such as LXML. An SSH library such as Paramiko may assist in the development of tools that can be used remotely.

The CLI does not support regular expressions. If you want to use regular expressions, you should use them in your script that calls the CLI.

2.10 Configuring the Oracle VM CLI

When Oracle VM Manager starts, it reads the CLI configuration file. The configuration file is located at:

`/u01/app/oracle/ovm-manager-3/ovm_cli/config/CLIConfigParams.xml`

You can change a number of options in the configuration file. These options are listed in this section.

Before you change any options in the configuration file, you should back up the original and change the permissions of the file to make it writeable:

```
# chmod +w /u01/app/oracle/ovm-manager-3/ovm_cli/config/CLIConfigParams.xml
```

Any changes to this configuration file require a restart of the CLI. See Section 2.1, “Starting and Stopping the CLI” for information on restarting the CLI.

2.10.1 SSH Port

By default, SSH connections to the CLI are allowed on port 10000. To change the port on which the CLI accepts connections, edit the `sshPort="10000"` line in the configuration file and change it to the port number you require.

2.10.2 Timeout

By default, connections to the CLI time out after 30 minutes. To change the timeout period for connections to the CLI client, edit the `clientInactivityTimeout="30"` line in the configuration file. Set the time out to be between 1-60 minutes. For security reasons, there is no option to disable the timeout period. Setting the value to less than 1 or greater than 60 causes the timeout value to revert to the default of 30 minutes.

In addition to this timeout option, you can use the ssh ServerAliveInterval option to stop an ssh client from timing out before the set timeout period. See the Tip in Section 2.2, “Connecting to the CLI” for more information on using this option.

2.10.3 Case Sensitivity

CLI commands are *not* case sensitive, so you can enter `list vm`, `List VM`, or any other variation in case.

Data values you use for objects in the CLI *are* case sensitive, so if a virtual machine has a name of `MyVM`, then you must use the same case when identifying it in the CLI and cannot use variations such as `myvm`.

2.10.4 Public Key Authentication Expiry

The connection channel for public key authentication expires after a designated period of time, or if Oracle VM Manager is restarted. The default for keeping the channel open is 1 week (10080 minutes). You can modify this setting by editing the `publicKeyAuthChannelTimeout="10080"` option in the configuration file. A value of `-1` keeps the channel open indefinitely.

2.10.5 Lock Exceptions

When executing multiple concurrent CLI scripts, you may experience lock exceptions around shared resources such as file systems and storage repositories. By default, connections to the CLI time out after 20 seconds if an object that is being requested by the CLI command is locked. If a command fails due to a lock exception, the command is resubmitted 12 times to see if the lock has become free and the command can be executed. To help you tune the CLI to reduce lock exceptions, you can configure the amount of time a command waits for the lock to become free, and the number of times the command is resubmitted with the following options in the configuration file:

- The lock time out period is defined by the `lockExceptionRetryInterval="20"` option. This option must be a value between 6 and 119 seconds, with the default being 20.
- The number of retry attempts is defined by the `lockExceptionRetryCount="12"` option. This option must be greater than 0.

2.11 CLI Logs

A log of the CLI application is available in:

`/u01/app/oracle/ovm-manager-3/domains/ovm_domain/servers/AdminServer/logs/CLI.log`

A log of commands submitted to the CLI is available in:

`/u01/app/oracle/ovm-manager-3/domains/ovm_domain/servers/AdminServer/logs/CLIAudit.log`

This log contains the following information about the command submitted to the CLI:

- Timestamp
- Client IP Address
- Username
- Command

The log files are rotated when the file size reaches 5 MB with up to 10 rotations, in the same way as the other Oracle VM log files.

Part I Using the CLI

The examples used in this section follow closely the format and flow used in the *Oracle VM Manager Getting Started Guide*. We have provided an abbreviated version of the steps and commands you need to get you started with using the CLI. If you need more information about the steps you are performing in this part, see the corresponding section in the *Oracle VM Manager Getting Started Guide* for overview information. If you need more detailed information on a step, see the *Oracle VM Manager User's Guide*.

Chapter 3 Discovering Oracle VM Servers

This section gives you the syntax and examples to discover an Oracle VM Server.

3.1 Discovering an Oracle VM Server

To discover an Oracle VM Server use the following syntax:

`discoverServer` `ipAddress=`*`value`* `password=`*`value`* `takeOwnership=` { `Yes` | `No` }

For example:

```
OVM> discoverServer ipAddress=10.172.76.73 password=password takeOwnership=Yes
```

For more information on the syntax and usage, see Section A.58, “discoverServer”.

Chapter 4 Discovering Storage

This section gives you the syntax and examples to use to *discover* storage using the CLI.

4.1 Discovering a file server

1. To discover a file server use the following syntax:

   `create FileServer plugin=`*`value`* `accessHost=`*`value`* `adminServers=`*`value`* [`refreshServers=`*`value`*] [`uniformedExports=` { `Yes` | `No` }] [`name=`*`value`*] [`description=`*`value`*]

   For example:

   ```
   OVM> create FileServer plugin="Oracle Generic Network File System" \
     accessHost=10.172.76.125 adminServers="MyServer1.virtlab.info,MyServer2.virtlab.info" \
     name=MyNFSServer
   ```

   For more information on the syntax and usage, see Section A.36, “create FileServer” .

2. After discovering a file server, you should attach an *admin server* to it so that administrative operations can be performed on the file server by Oracle VM Manager. To add an admin server to the file server, use the syntax:

   `addAdminServer` { `FileServer` | `StorageArray` } *`instance`* `server=`*`value`*

   For example:

   ```
   OVM> addAdminServer FileServer name=MyNFSServer server=MyServer
   ```

   For more information on the syntax and usage, see Section A.4, “addAdminServer” .

3. You should also attach a refresh server to the file server which is used to refresh the file system. To add a refresh server to the file server, use the syntax:

   `addRefreshServer FileServer` *`instance`* `server=`*`value`*

   For example:

   ```
   OVM> addRefreshServer FileServer name=MyNFSServer server=MyServer
   ```

   For more information on the syntax and usage, see Section A.6, “addRefreshServer” .

4. The final step is to refresh the file server so Oracle VM Manager has the most current information about the file server. To refresh the file server, use the syntax:

   `refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

   For example:

   ```
   OVM> refresh FileServer name=MyNFSServer
   ```

   For more information on the syntax and usage, see Section A.124, “refresh” .

4.2 Discovering a storage array

1. To discover a storage array use the following syntax:

`create StorageArray` `plugin=`*`value`* `storageType=` { `FIBRECHANNEL` | `ISCSI` | `LOCAL` | `UNKNOWN` } [`storageName=`*`value`*] [`accessHost=`*`value`*] [`accessPort=`*`value`*] [`accessUsername=`*`value`* `accessPassword=`*`value`*] [`useChap=` { `Yes` | `No` }] [`adminHost=`*`value`* `adminUserName=`*`value`* `adminPassword=`*`value`*] [`pluginPrivateData=`*`value`*] `name=`*`value`* [`description=`*`value`*]

For example:

```
OVM> create StorageArray plugin="Oracle Generic SCSI Plugin" name=MyISCSIServer \
  storageType=ISCSI accessHost=10.172.76.130 accessPort=3260
```

For more information on the syntax and usage, see Section A.47, “create StorageArray” .

2. After discovering a storage array, you should attach at least one *admin server* to it so that administrative operations can be performed on the storage array by Oracle VM Manager. To add an admin server to the storage array, use the syntax:

   `addAdminServer` { `FileServer` | `StorageArray` } *`instance`* `server=`*`value`*

   For example:

   ```
   OVM> addAdminServer StorageArray name=MyISCSIServer server=MyServer
   ```

   For more information on the syntax and usage, see Section A.4, “addAdminServer” .

3. Next you should add the storage initiators to an access group for each Oracle VM Server that is to be granted access to the storage. In this example we add the storage initiators for each Oracle VM Server to the default access group that is created when a storage array is discovered. First, find the name of the access group by listing the access groups for the server using the `show StorageArray` command, for example:

   ```
   OVM> show StorageArray name=MyISCSIServer
     ...
     Access Group 1 = Default access group @ MyISCSIServer @ 0004fb00000900005264cefc5b9a1cb8
       [Default access group @ MyISCSIServer]  Access group name
     Storage Plug-in = oracle.generic.SCSIPlugin.GenericPlugin (1.1.0)
       [Oracle Generic SCSI Plugin]
     ...
   ```

   Next, find the storage initiator name for each Oracle VM Server using the `show Server` command, for example:

   ```
   OVM> show Server name=MyServer1
      ...
     Storage Initiator 1 = iqn.1988-12.com.oracle:e774e056fd3
            Storage initiator name
     Storage Initiator 2 = storage.LocalStorageInitiator in
       00:e0:81:4d:40:f5:00:e0:81:4d:40:be:00:e0:81:4d
     ...
   ```

   Then add the storage initiator for each Oracle VM Server to the default access group using the syntax:

   `add StorageInitiator` *`instance`* `to AccessGroup` *`instance`*

   For example:

   ```
   OVM> add StorageInitiator name=iqn.1988-12.com.oracle:d72d82d0817f to AccessGroup \
     name='Default access group @ MyISCSIServer'
   ```

   For more information on the syntax and usage, see Section A.13, “add StorageInitiator” .

4. Validate the storage array. Validation should be performed when any changes are made to the storage. To validate the storage array, use the syntax:

   `validate StorageArray` *`instance`*

   For example:

   ```
   OVM> validate StorageArray name=MyISCSIServer
   ```

   For more information on the syntax and usage, see Section A.161, “validate” .

5. Refresh the storage array so Oracle VM Manager has the most current information about the storage. To refresh the storage array, use the syntax:

   `refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

   For example:

   ```
   OVM> refresh StorageArray name=MyISCSIServer
   ```

   For more information on the syntax and usage, see Section A.124, “refresh” .

Chapter 5 Creating a Network

This section gives you the syntax and examples to create a network with the Virtual Machine role.

5.1 Creating a virtual machine network

1. Create an Ethernet-based network with the Virtual Machine role using the syntax:

   `create Network` [`roles=` { `MANAGEMENT` | `LIVE_MIGRATE` | `CLUSTER_HEARTBEAT` | `VIRTUAL_MACHINE` | `STORAGE` }] `name=`*`value`* [`description=`*`value`*] [`on Server` *`instance`*]

   For example:

   ```
   OVM> create Network name=MyVMNetwork roles=VIRTUAL_MACHINE
   ```

   For more information on the syntax and usage, see Section A.38, "create Network".

2. Next, find an Ethernet port from each Oracle VM Server to add to the network. First, find the ID of an Ethernet port using the `show Server` command, for example:

   ```
   OVM> show Server name=MyServer1
     ...
     Ethernet Port 1 = 0004fb00002000007711332ff75857ee  [eth0 on MyServer3.virtlab.info]
     Ethernet Port 2 = 0004fb0000200000d2e7d2d352a6654e  [eth1 on MyServer3.virtlab.info]
     Ethernet Port 3 = 0004fb0000200000c12192a08f2236e4  [eth2 on MyServer3.virtlab.info]
     ...
   OVM>
   ```

   Then, add a port from each Oracle VM Server to the network using the syntax:

   `add Port` *`instance`* `to` { `BondPort` | `Network` } *`instance`*

   For example:

   ```
   OVM> add Port id=0004fb0000200000d2e7d2d352a6654e to Network name=MyVMNetwork
   ```

   For more information on the syntax and usage, see Section A.10, "add Port".

Chapter 6 Creating a Server Pool

This section gives you the syntax and examples to create a server pool and add Oracle VM Server to it.

6.1 Creating a server pool

1. If you are creating a clustered server pool you must provide a file system or physical disk to use for the server pool file system. To find a file system or physical disk, use the list command, for example:

```
OVM> list FileSystem
  id:66a61958-e61a-44fe-b0e0-9dd64abef7e3  name:nfs on 10.172.76.125:/mnt/vol1/poolfs03
  id:0004fb0000050000b85745f78b0c4b61  name:fs on 350014ee2568cc0cf
  id:4ebb1575-e611-4662-87b9-a84b40ce3db7  name:nfs on 10.172.76.125:/mnt/vol1/poolfs04
  id:858d98c5-3d8b-460e-9160-3415cbdda738  name:nfs on 10.172.76.125:/mnt/vol1/poolfs01
  id:0dea4818-20e6-4d3a-958b-b12cf91588b5  name:nfs on 10.172.76.125:/mnt/vol1/poolfs02
  id:35b4f1c6-182b-4ea5-9746-51393f3b515c  name:nfs on 10.172.76.125:/mnt/vol2/repo03
  id:aeb6143d-0a96-4845-9690-740bbf1e225e  name:nfs on 10.172.76.125:/mnt/vol1/repo01
  id:05e8536f-8d9c-4d7c-bbb2-29b3ffafe011  name:nfs on 10.172.76.125:/mnt/vol2/repo02
  id:0004fb00000500006a46a8dbd2461939  name:MyServerPool_cluster_heartbeat
  id:0004fb00000500000809e28f4fab56b1  name:fs on 350014ee20137ee44
OVM> list PhysicalDisk
  id:0004fb000018000019b86ccf3f473a9e  name:FreeBSD (9)
  id:0004fb0000180000c4609a67d55b5803  name:FreeBSD (3)
  id:0004fb00001800002179de6afe5f0cf3  name:SATA_WDC_WD5001ABYS-_WD-WCAS86288968
  id:0004fb0000180000a0b43f9684fc78ac  name:FreeBSD (2)
  id:0004fb0000180000732be086afb26911  name:FreeBSD (7)
  id:0004fb000018000067ce80973e18374e  name:FreeBSD (8)
  id:0004fb000018000035ce16ee4d58dc4d  name:FreeBSD (1)
  id:0004fb00001800006855117242d9a537  name:FreeBSD (6)
  id:0004fb0000180000a9c7a87ba52ce5ec  name:FreeBSD (5)
  id:0004fb0000180000ebabef9838188d78  name:SATA_WDC_WD5001ABYS-_WD-WCAS86571931
  id:0004fb00001800008f6ea92426f2cfb8  name:SATA_WDC_WD5001ABYS-_WD-WCAS86257005
  id:0004fb00001800008ccb1925cdbbd181  name:SATA_WDC_WD5001ABYS-_WD-WCAS86578538
  id:0004fb0000180000e034b4662665161c  name:FreeBSD (4)
```

   Before you create a clustered server pool you must refresh the file system or physical disk to be used for the server pool file system. To refresh a file system, use the syntax:

   `refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

   For example, to refresh a physical disk:

```
OVM> refresh PhysicalDisk id=0004fb000018000035ce16ee4d58dc4d
```

   And to refresh a file system:

```
OVM> refresh FileSystem name="nfs on 10.172.76.125://mnt//vol1//repo01"
```

   For more information on the syntax and usage, see Section A.124, "refresh" .

2. To create a server pool use the following syntax:

   `create ServerPool` `virtualIP=`*`value`* `clusterEnable=` { `Yes` | `No` } [`clusterTimeout=`*`value`*] [`fileSystem=`*`value`*] [`physicalDisk=`*`value`*] [`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }] [`migrateUsingSsl=` { `Yes` | `No` }] [`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` }] [`policyMode=` { `OFF` | `DRS` | `DPM` }] [`policyCpuEnable=` { `Yes` | `No` }] [`policyPeriod=`*`value`*] [`policyCpuThreshold=`*`value`*] `name=`*`value`* [`description=`*`value`*]

For example to create a clustered server pool:

```
OVM> create ServerPool virtualIP=10.172.77.195 clusterEnable=Yes \
  filesystem="nfs on 10.172.76.125://mnt//vol1//poolfs01" name=MyServerPool \
  description='Clustered server pool'
```

And to create an unclustered server pool:

```
OVM> create ServerPool virtualIP=10.172.77.195 clusterEnable=No name=MyServerPool \
  description='Unclustered server pool'
```

For more information on the syntax and usage, see Section A.43, “create ServerPool”.

6.2 Adding Oracle VM Servers to a server pool

To add Oracle VM Servers to a server pool use the following syntax:

`add Server` *`instance`* `to` { `AccessGroup` | `CpuCompatibilityGroup` | `Repository` | `ServerPool` } *`instance`*

For example:

```
OVM> add Server name=MyServer to ServerPool name=MyServerPool
```

For more information on the syntax and usage, see Section A.11, “add Server” .

Chapter 7 Creating a Storage Repository

This section gives you the syntax and examples to use to create a storage repository.

7.1 Creating a storage repository on a file server

1. Find the file system you want to use to create the storage repository with the list command, for example:

   ```
   OVM> list FileSystem
     id:66a61958-e61a-44fe-b0e0-9dd64abef7e3  name:nfs on 10.172.76.125:/mnt/vol1/poolfs03
     id:0004fb0000050000b85745f78b0c4b61  name:fs on 350014ee2568cc0cf
     id:4ebb1575-e611-4662-87b9-a84b40ce3db7  name:nfs on 10.172.76.125:/mnt/vol1/poolfs04
     id:858d98c5-3d8b-460e-9160-3415cbdda738  name:nfs on 10.172.76.125:/mnt/vol1/poolfs01
     id:0dea4818-20e6-4d3a-958b-b12cf91588b5  name:nfs on 10.172.76.125:/mnt/vol1/poolfs02
     id:35b4f1c6-182b-4ea5-9746-51393f3b515c  name:nfs on 10.172.76.125:/mnt/vol2/repo03
     id:aeb6143d-0a96-4845-9690-740bbf1e225e  name:nfs on 10.172.76.125:/mnt/vol1/repo01
     id:05e8536f-8d9c-4d7c-bbb2-29b3ffafe011  name:nfs on 10.172.76.125:/mnt/vol2/repo02
     id:0004fb00000500006a46a8dbd2461939  name:MyServerPool_cluster_heartbeat
     id:0004fb00000500000809e28f4fab56b1  name:fs on 350014ee20137ee44
   OVM>
   ```

   Then, refresh the file system you intend to use for the storage repository. To refresh the file system, use the syntax:

   `refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

   For example:

   ```
   OVM> refresh FileSystem name="nfs on 10.172.76.125://mnt//vol1//repo01"
   ```

   For more information on the syntax and usage, see Section A.124, “refresh” .

2. Create the storage repository. Use the syntax:

   `create Repository` [`sharePath=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on FileSystem` *`instance`*

   For example:

   ```
   OVM> create Repository name=MyRepository on FileSystem \
     name="nfs on 10.172.76.125://mnt//vol2//repo01"
   ```

   For more information on the syntax and usage, see Section A.40, “create Repository” .

3. To grant access to the storage repository to a server pool, you must *present* the repository. To present the storage repository to server pool, use the syntax:

   `add ServerPool` *`instance`* `to` { `AccessGroup` | `Repository` } *`instance`*

   For example:

   ```
   OVM> add ServerPool name=MyServerPool to Repository name=MyRepository
   ```

   For more information on the syntax and usage, see Section A.12, “add ServerPool” .

4. Finally, refresh the storage repository using the syntax:

`refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

For example:

```
OVM> refresh Repository name=MyRepository
```

For more information on the syntax and usage, see Section A.124, “refresh” .

7.2 To create a storage repository on a storage array

1. Find the physical disk (LUN) you want to use to create the storage repository with the list command, for example:

```
OVM> list PhysicalDisk
  id:0004fb000018000067ce80973e18374e  name:MyLUN1
  id:0004fb000018000035ce16ee4d58dc4d  name:MyLUN2
  id:0004fb0000180000a9c7a87ba52ce5ec  name:MyLUN3
OVM>
```

2. Find a local file system on an Oracle VM Server that has access to the LUN with the list command, for example:

```
OVM> list FileServer
  id:0004fb0000090000d773cb3fe655865a  name:Local FS MyServer1
  id:0004fb000009000014baa666cdf62317  name:Local FS MyServer2
  id:0004fb00000900008ae3eb6203f5646c  name:MyNFSStorage
  id:0004fb00000900001d523e2a6ce1f8c8  name:Local FS MyServer3
OVM>
```

3. Create an OCFS2 file system on the LUN using the local file system on the Oracle VM Server as the FileServer. Use the syntax:

`create FileSystem` `physicalDisk=`*`value`* `name=`*`value`* [`description=`*`value`*] `on` `FileServer` *`instance`*

For example:

```
OVM> create FileSystem name=MyRepoFileSystem physicalDisk="MyLUN1" on FileServer \
  name="Local FS MyServer1"
```

For more information on the syntax and usage, see Section A.37, “create FileSystem” .

4. Create the storage repository. Use the syntax:

`create Repository` [`sharePath=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on` `FileSystem` *`instance`*

For example:

```
OVM> create Repository name=MyRepository on FileSystem name=MyRepoFileSystem
```

For more information on the syntax and usage, see Section A.40, “create Repository” .

5. To grant access to the storage repository to a server pool, you must *present* the repository. To present the storage repository to server pool, use the syntax:

`add ServerPool` *`instance`* `to` { `AccessGroup` | `Repository` } *`instance`*

For example:

```
OVM> add ServerPool name=MyServerPool to Repository name=MyRepository
```

For more information on the syntax and usage, see Section A.12, “add ServerPool” .

6. Finally, refresh the storage repository using the syntax:

   `refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

   For example:

   ```
   OVM> refresh Repository name=MyRepository
   ```

   For more information on the syntax and usage, see Section A.124, “refresh” .

7.3 Importing resources to a storage repository

7.3.1 Importing an assembly

To import an assembly to the storage repository, use the following syntax:

`importAssembly` `Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

For example:

```
OVM> importAssembly Repository name=MyRepository url="http:////example.com//myassembly.ova"
```

For more information on the syntax and usage, see Section A.115, “importAssembly”.

7.3.2 Importing a virtual machine template

To import a virtual machine template to the storage repository, use the following syntax:

`importTemplate` `Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

For example:

```
OVM> importTemplate Repository name=MyRepository url="http:////example.com//mytemplate.tgz"
```

For more information on the syntax and usage, see Section A.116, “importTemplate”.

7.3.3 Importing an ISO file

To import an ISO file to the storage repository, use the following syntax:

`importVirtualCdrom` `Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

For example:

```
OVM> importVirtualCdrom Repository name=MyRepository url="http:////example.com//myiso.iso"
```

For more information on the syntax and usage, see Section A.117, “importVirtualCdrom”.

Chapter 8 Creating a Virtual Machine

This section gives you the syntax and examples to use to create a virtual machine from a number of sources.

8.1 Creating a virtual machine from a template

Clone a virtual machine from a template, using the syntax:

`clone Vm` *`instance`* `destType=` { `Vm` | `VmTemplate` } [`destName=`*`value`*] `serverPool=`*`value`* [`cloneCustomizer=`*`value`*] [`targetRepository=`*`value`*]

For example:

```
OVM> clone Vm name=MyVMTemplate.tgz destType=Vm destName=MyNewVM serverPool=MyServerPool
```

For more information on the syntax and usage, see Section A.24, “clone Vm” .

8.2 Creating a virtual machine from an assembly

1. When you import an assembly file, each virtual machine in the assembly file is unpacked and stored as an AssemblyVm object. The AssemblyVm object is then used to create a virtual machine template (which in turn is then used to create a virtual machine), not the assembly file itself. Use the `show Assembly` command to find the name or ID of the new AssemblyVm objects in an assembly.

   ```
   OVM> show assembly name=myassembly.ova
     Origin = http://example.com/myassembly.ova
     Repository = 0004fb00000300007f6b93d990fd718c  [MyRepository]
     Assembly Vm 1 = 11a8af41a2_vm_OVM_OL6U1_x86_64_PVHVM ID [MyVM] Name
     Assembly VirtualDisk 1 = 11a8af41a2_disk_system  [system]
     Id = 11a8af41a2  [myassembly.ova]
     Name = myassembly.ova
     Description = Import URL: http://example.com/myassembly.ova
     Locked = false
   ```

2. Create a virtual machine template from an AssemblyVm object using the syntax:

   `createVmFromAssembly AssemblyVm` *`instance`*

   Make sure you use the name of the AssemblyVm object created when you imported the assembly file, not the name of the assembly file.

   For example:

   ```
   OVM> createVmFromAssembly AssemblyVm name=myassembly.ova
   ```

   For more information on the syntax and usage, see Section A.31, “createVmFromAssembly” .

3. The previous step creates a virtual machine template with the name *`assembly_name`*_*`vm_name`*, for example:

   myassembly.ova_myvm

   You can see a list of virtual machines (and templates) using the `list Vm` command, then use the `show Vm` command to get the name or ID of the new template. Clone the new virtual machine template to a virtual machine as shown in Section 8.1, “Creating a virtual machine from a template”.

8.3 Creating a virtual machine from an ISO

1. Create a virtual machine using the syntax:

   `create Vm` [`memory=`*`value`*] [`memoryLimit=`*`value`*] [`cpuCount=`*`value`*] [`cpuCountLimit=`*`value`*] [`cpuPriority=`*`value`*] [`cpuUtilizationCap=`*`value`*] [`highAvailability=` { `Yes` | `No` }] [`hugePagesEnabled=` { `Yes` | `No` }] [`osType=`*`value`*] [`mouseType=` { `OS_DEFAULT` | `PS2_MOUSE` | `USB_MOUSE` | `USB_TABLET` }] `domainType=` { `XEN_HVM` | `XEN_HVM_PV_DRIVERS` | `XEN_PVM` | `LDOMS_PVM` | `UNKNOWN` } [`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }] [`bootOrder=` { `PXE` | `DISK` | `CDROM` }] [`networkInstallPath=`*`value`*] `repository=`*`value`* [`server=`*`value`*] [`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` | `USE_POOL_POLICY` }] `name=`*`value`* [`description=`*`value`*] `on ServerPool` *`instance`*

   For example:

   ```
   OVM> create Vm name=MyVM repository=MyRepository domainType=XEN_HVM \
     server=MyServer startPolicy=USE_POOL_POLICY on ServerPool name=MyServerPool
   ```

   For more information on the syntax and usage, see Section A.51, “create Vm” .

2. Create a virtual disk to use as the boot disk using the syntax:

   `create VirtualDisk size=`*`value`* `shareable=` { `Yes` | `No` } `sparse=` { `Yes` | `No` } `name=`*`value`* [`description=`*`value`*] `on Repository` *`instance`*

   For example:

   ```
   OVM> create VirtualDisk name=MyVMDisk size=10 sparse=Yes shareable=No on Repository \
     name=MyRepository
   ```

   For more information on the syntax and usage, see Section A.49, “create VirtualDisk” .

3. Map the virtual disk to the virtual machine using the syntax:

   `create VmDiskMapping slot=`*`value`* { `physicalDisk=`*`value`* | `virtualDisk=`*`value`* | `virtualCd=` { *`value`* | `EMPTY_CDROM` } } `name=`*`value`* [`description=`*`value`*] `on Vm` *`instance`*

   For example:

   ```
   OVM> create VmDiskMapping slot=0 virtualDisk=MyVMDisk name="Boot Disk" on Vm name=MyVM
   ```

   For more information on the syntax and usage, see Section A.55, “create VmDiskMapping” .

4. Map an ISO file to the virtual machine using the syntax:

   `create VmDiskMapping slot=`*`value`* { `physicalDisk=`*`value`* | `virtualDisk=`*`value`* | `virtualCd=` { *`value`* | `EMPTY_CDROM` } } `name=`*`value`* [`description=`*`value`*] `on Vm` *`instance`*

   For example:

   ```
   OVM> create VmDiskMapping slot=1 virtualCd=OracleLinux-dvd.iso \
     name="CDROM Drive" on Vm name=MyVM
   ```

   For more information on the syntax and usage, see Section A.55, “create VmDiskMapping”.

5. Set up the disk boot order as the CDROM (ISO file) as the first disk, then the virtual disk as the secondary disk, using the syntax:

`edit Vm` *`instance`* [`memory=`*`value`*] [`memoryLimit=`*`value`*] [`cpuCount=`*`value`*] [`cpuCountLimit=`*`value`*] [`cpuPriority=`*`value`*] [`cpuUtilizationCap=`*`value`*] [`highAvailability=` { `Yes` | `No` }] [`hugePagesEnabled=` { `Yes` | `No` }] [`osType=`*`value`*] [`mouseType=` { `OS_DEFAULT` | `PS2_MOUSE` | `USB_MOUSE` | `USB_TABLET` }] [`domainType=` { `XEN_HVM` | `XEN_HVM_PV_DRIVERS` | `XEN_PVM` | `LDOMS_PVM` | `UNKNOWN` }] [`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }] [`bootOrder=` { `PXE` | `DISK` | `CDROM` }] [`networkInstallpath=`*`value`*] [`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` | `USE_POOL_POLICY` }] [`name=`*`value`*] [`description=`*`value`*]

For example:

```
OVM> edit Vm name=MyVM bootOrder='CDROM,DISK' startPolicy=BEST_SERVER
```

For more information on the syntax and usage, see Section A.90, “edit Vm”.

6. Create a VNIC and add it to the virtual machine using the syntax:

   `create Vnic` [`macAddress=`*`value`*] `network=`*`value`* `name=`*`value`* [`description=`*`value`*] `on Vm` *`instance`*

   For example:

   ```
   OVM> create Vnic name=00:21:f6:00:00:18 network=MyVMNetwork on Vm name=MyVM
   ```

   For more information on the syntax and usage, see Section A.56, “create Vnic”.

8.4 To start a virtual machine:

Start the virtual machine, using the syntax:

`start` { `Server` | `Vm` } *`instance`*

For example:

```
OVM> start Vm name=MyVM
```

For more information on the syntax and usage, see Section A.156, “start” .

Part II CLI Command Reference

This part gives the full syntax of each CLI command with examples.

In some commands such as any command that edits an object, you may see slight differences between the syntax in the CLI syntax help, and that documented here. This is because the CLI syntax help uses an asterisk to mark options that are mandatory for an object and maps directly to mandatory options when using the Oracle VM Manager user interface, but not necessarily mandatory when entering a command in the CLI. The syntax documented in this section instead shows you what is optional or mandatory when using that command in the CLI.

Appendix A CLI Command Reference

This appendix gives the full syntax of each CLI command, with usage examples.

A.1 abort Job

Aborts a job.

Syntax

`abort Job` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command aborts a running job.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyJob`.

Examples

Example A.1 Aborting a job

```
OVM> abort job id=1342399962239
```

See Also

- Section A.101, “getDebugTranscript”
- Section A.105, “getJobs”
- Section A.121, “list”
- Section A.151, “show”

A.2 ackEvent

Acknowledges an event.

Syntax

`ackEvent eventId=`*`value`*

Description

This command acknowledges an event.

Options

The following table shows the available options for this command.

Option	Description
`eventId=value`	The ID of the event. To get the ID of an event, use the getEvents or getEventsForObject command.

Examples

Example A.2 Acknowledging an event

```
OVM> ackEvent eventId=1342155856562
```

See Also

- Section A.103, "getEvents"
- Section A.104, "getEventsForObject"

A.3 addAccessHost

Adds an access host to an ISCSI server.

Syntax

`addAccessHost StorageArray` *`instance`* `accessHost=`*`value`* [`accessPort=`*`value`*] [`accessUsername=`*`value`* `accessPassword=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds an access host to an ISCSI storage array. Adding more than one access host provides multiple network paths to the storage. Create an access host for each path when using *multipathing*. At least one access host must be set. This is not applicable to fibre channel storage. To remove an access host, use the removeAccessHost command.

Options

The following table shows the available options for this command.

Option	Description
`accessHost=value`	The hostname or IP address for the access host.
`accessPort=value`	The port on which to connect to the access host. The default port of `3260` is used if no value is provided.
`accessUsername=value`	The username to use when using CHAP authentication.
`accessPassword=value`	The password for the `accessUsername` user.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyISCSIServer`.

Examples

Example A.3 Adding a storage array access host

```
OVM> addAccessHost StorageArray name=MyISCSIServer accessHost=10.172.76.131
```

See Also

- Section A.128, “removeAccessHost”
- Section A.47, “create StorageArray”
- Section A.84, “edit StorageArray”
- Section A.121, “list”
- Section A.151, “show”

A.4 addAdminServer

Adds an administrative Oracle VM Server to a file server or storage array.

Syntax

`addAdminServer` { `FileServer` | `StorageArray` } *`instance`* `server=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds an administrative Oracle VM Server to a file server or storage array.

Options

The following table shows the available options for this command.

Option	Description
`server=`*`value`*	The name or ID of the administrative Oracle VM Server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyFileServer`.

Examples

Example A.4 Adding an admin server to a file server

```
OVM> addAdminServer FileServer name=MyNFSServer server=MyServer
```

Example A.5 Adding an admin server to a storage array

```
OVM> addAdminServer StorageArray name=MyISCSIServer server=MyServer
```

See Also

- Section A.129, “removeAdminServer”

- Section A.36, “create FileServer”
- Section A.47, “create StorageArray”
- Section A.68, “edit FileServer”
- Section A.84, “edit StorageArray”
- Section A.136, “remove Server”
- Section A.11, “add Server”
- Section A.6, “addRefreshServer”
- Section A.131, “removeRefreshServer”
- Section A.84, “edit StorageArray”

A.5 addPolicyServer

Adds a policy server to a server pool.

Syntax

`addPolicyServer` `ServerPool` *`instance`* `server=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds a policy server to a server pool.

Options

The following table shows the available options for this command.

Option	Description
`server=`*`value`*	The name or ID of an Oracle VM Server to add as a policy server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.6 Adding a policy server to a server pool

```
OVM> addPolicyServer ServerPool name=MyServerPool server=MyServer
```

See Also

- Section A.130, “removePolicyServer”
- Section A.44, “create ServerPoolNetworkPolicy”

- Section A.81, “edit ServerPoolNetworkPolicy”

A.6 addRefreshServer

Adds a *refresh server* to a file server.

Syntax

`addRefreshServer FileServer` *`instance`* `server=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds a refresh server to a file server. The refresh server is an Oracle VM Server that is used to refresh the file systems on an NFS file server. You can add multiple refresh servers to a file server. A file server must have at least one refresh server assigned to it.

Options

The following table shows the available options for this command.

Option	**Description**
`server=`*`value`*	The name or ID of the Oracle VM Server to be used as a refresh server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.7 Adding a refresh server to file server

```
OVM> addRefreshServer FileServer name=MyNFSServer server=MyServer
```

See Also

- Section A.131, “removeRefreshServer”
- Section A.36, “create FileServer”
- Section A.68, “edit FileServer”
- Section A.136, “remove Server”
- Section A.11, “add Server”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.124, “refresh”

A.7 add BondPort

Adds an bonded port to a network object.

Syntax

`add BondPort` *`instance`* `to Network` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds a bonded port to a network object.

Options

The following table shows the available options for this command.

Option	**Description**
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyNetwork`.

Examples

Example A.8 Adding a bonded port to a network

```
OVM> add BondPort id=0004fb000020000065822cb7bb9ec296 to Network name=MyVMNetwork
```

See Also

- Section A.34, “create BondPort”
- Section A.64, “edit BondPort”
- Section A.132, “remove BondPort”
- Section A.135, “remove Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.8 add FileSystem

Adds a file system to an access group.

Syntax

`add FileSystem` *`instance`* `to AccessGroup` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Note that if the instance name contains forward slashes, these must be escaped using an additional forward slash. This is illustrated in the examples for this command.

Description

This command adds a file system to an access group.

A file system may only be associated with one access group. If you create a new access group for a file system that is already associated with an existing access group, the file system is disassociated from the original access group.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAccessGroup` .

Examples

Example A.9 Adding a file system to an access group

```
OVM> add FileSystem name="nfs on 10.172.76.125://mnt//vol2//repo03" to AccessGroup \
  name=MyAccessGroup
```

See Also

- Section A.37, “create FileSystem”
- Section A.70, “edit FileSystem”
- Section A.133, “remove FileSystem”
- Section A.32, “create AccessGroup”
- Section A.59, “edit AccessGroup”
- Section A.121, “list”
- Section A.151, “show”
- Section A.57, “delete”

A.9 add PhysicalDisk

Adds a physical disk to a SAN storage access group.

Syntax

`add PhysicalDisk` *`instance`* `to AccessGroup` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds a physical disk to a SAN storage access group. *Local storage* and generic storage plug-ins are not supported with this command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAccessGroup`.

Examples

Example A.10 Adding a physical disk to a SAN storage access group

```
OVM> add PhysicalDisk id=0004fb00001800007ee6dbda7b4461cb to AccessGroup \
  name='Default access group @ MyISCSIServer'
```

See Also

- Section A.39, “create PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.134, “remove PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.10 add Port

Adds an Ethernet port to a network object.

Syntax

`add Port` *`instance`* `to` { `BondPort` | `Network` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds an Ethernet port to a network object.

To configure the IP address for a port, use the embeddedCreate command.

Options

The following table shows the available options for this command.

Option	Description
{ `BondPort` | `Network` }	The network object to which to add the Ethernet port.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyNetwork`.

Examples

Example A.11 Adding an Ethernet port to a network

```
OVM> add Port id=0004fb0000200000d2e7d2d352a6654e to Network name=MyVMNetwork
```

See Also

- Section A.75, “edit Port”
- Section A.135, “remove Port”
- Section A.38, “create Network”
- Section A.72, “edit Network”
- Section A.34, “create BondPort”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.121, “list”
- Section A.151, “show”

A.11 add Server

Adds an Oracle VM Server to an object.

Syntax

`add Server` *`instance`* `to` { `AccessGroup` | `CpuCompatibilityGroup` | `Repository` | `ServerPool` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds an Oracle VM Server to either a *CPU compatibility group*, server pool, storage repository or access group.

When you add an Oracle VM Server to a storage repository, you are making that Oracle VM Server available to perform admin duties for that storage object.

To present a storage repository to all Oracle VM Servers in a server pool, use the add ServerPool command.

To add admin servers to a file server or storage array, use the addAdminServer command.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `CpuCompatibilityGroup` | `Repository` | `ServerPool` }	The object on which to assign the Oracle VM Server as an admin server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.12 Adding an Oracle VM Server to a CPU compatibility group

```
OVM> add Server name=MyServer to CpuCompatibilityGroup name=MyCPUGroup
```

Example A.13 Adding an Oracle VM Server to a server pool

```
OVM> add Server name=MyServer to ServerPool name=MyServerPool
```

Example A.14 Adding an Oracle VM Server to an access group

```
OVM> add Server name=MyServer to AccessGroup name=MyAccessGroup
```

Example A.15 Adding (presenting) an Oracle VM Server to a storage repository

```
OVM> add Server name=MyServer to Repository name=MyRepository
```

See Also

- Section A.12, “add ServerPool”
- Section A.58, “discoverServer”
- Section A.78, “edit Server”
- Section A.124, “refresh”
- Section A.156, “start”
- Section A.157, “stop”
- Section A.144, “restart”
- Section A.120, “kill”
- Section A.160, “upgrade”
- Section A.4, “addAdminServer”

- Section A.129, “removeAdminServer”
- Section A.6, “addRefreshServer”
- Section A.131, “removeRefreshServer”
- Section A.136, “remove Server”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.12 add ServerPool

Adds a server pool to a storage repository or to an access group.

Syntax

`add ServerPool` *`instance`* `to` { `AccessGroup` | `Repository` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command presents a storage repository to all Oracle VM Servers in a server pool. To present a storage repository to an individual Oracle VM Server, use the add Server command.

This command also adds a server pool to an access group.

Important

The option to add or present an entire server pool is a convenience that automatically selects all of the servers that belong to the specified pool and then performs the action on those servers. There is no actual relationship between the server pool and the repository or access group stored within Oracle VM Manager. This means that if you add a server to a server pool after having presented a repository to the server pool, the repository is not automatically presented to the new server. Equally, removing a server from the server pool does not automatically update the configuration.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `Repository` }	The object on which to add or present the server pool.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.16 Presenting a storage repository to a server pool

```
OVM> add ServerPool name=MyServerPool to Repository name=MyRepository
```

Example A.17 Adding a server pool to an access group

```
OVM> add ServerPool name=MyServerPool to AccessGroup name=MyAccessGroup
```

See Also

- Section A.137, “remove ServerPool”
- Section A.11, “add Server”
- Section A.121, “list”
- Section A.151, “show”

A.13 add StorageInitiator

Adds a storage initiator to an access group for a SAN storage server.

Syntax

`add StorageInitiator` *`instance`* `to AccessGroup` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds a storage initiator to an access group for a SAN storage server.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAccessGroup`.

Examples

Example A.18 Adding a storage initiator

```
OVM> add StorageInitiator name=iqn.1988-12.com.oracle:d72d82d0817f to AccessGroup \
  name='Default access group @ MyISCSIServer'
```

See Also

- Section A.138, “remove StorageInitiator”
- Section A.121, “list”

- Section A.151, “show”

A.14 add Tag

Adds a tag to an object.

Syntax

`add Tag` *`instance`* `to` { `Server` | `ServerPool` | `Vm` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command adds a tag used to identify and group objects to an object.

Options

The following table shows the available options for this command.

Option	Description
{ `Server` | `ServerPool` | `Vm` }	The object on which to add the tag.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyTag`.

Examples

Example A.19 Adding a tag to a server pool

```
OVM> add Tag name=MyTag to ServerPool name=MyServerPool
```

See Also

- Section A.48, “create Tag”
- Section A.86, “edit Tag”
- Section A.139, “remove Tag”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.15 add VlanInterface

Adds a VLAN interface to a network.

Syntax

`add VlanInterface` *`instance`* `to` `Network` *`instance`*

Where *instance* is:

{ `id=`*value* | `name=`*value* }

Description

This command adds a VLAN interface to a network. To create a VLAN interface, use the create VlanInterface command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*value* | `name=`*value* }	The instance of the object using either the `id` or `name` option, for example `name=MyNetwork`.

Examples

Example A.20 Adding a VLAN interface to a network

```
OVM> add VlanInterface name=MyVLANInterface to Network name=MyNetwork
```

See Also

- Section A.50, “create VlanInterface”
- Section A.89, “edit VlanInterface”
- Section A.140, “remove VlanInterface”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.16 add Vm

Adds a virtual machine to an Oracle VM Server, server pool, or anti affinity group.

Syntax

`add Vm` *instance* `to` { `AntiAffinityGroup` | `Server` | `ServerPool` } *instance*

Where *instance* is:

{ `id=`*value* | `name=`*value* }

Description

This command adds a virtual machine to an Oracle VM Server, server pool, or anti affinity group. The virtual machine cannot be running, and must be stopped before using this command.

Options

The following table shows the available options for this command.

Option	Description
{ `AntiAffinityGroup` | `Server` | `ServerPool` }	The object on which to add the virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.21 Adding a virtual machine to a server pool

```
OVM> add Vm name=MyVM to ServerPool name=MyServerPool
```

Example A.22 Adding a virtual machine to an Oracle VM Server

```
OVM> add Vm name=MyVM to Server name=MyServer
```

Example A.23 Adding a virtual machine to an anti affinity group

```
OVM> add Vm name=MyVM to AntiAffinityGroup name=MyAAGroup
```

See Also

- Section A.51, “create Vm”
- Section A.119, “importVirtualMachine”
- Section A.33, “create AntiAffinityGroup”
- Section A.141, “remove Vm”
- Section A.122, “migrate Vm”
- Section A.24, “clone Vm”
- Section A.156, “start”
- Section A.151, “show”
- Section A.121, “list”

A.17 add Vnic

Adds a VNIC to a network.

Syntax

`add Vnic` *`instance`* `to Network` *`instance`*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command adds a VNIC to a network.

Options

The following table shows the available options for this command.

Option	Description
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyNetwork.

Examples

Example A.24 Adding a VNIC to a network

```
OVM> add Vnic name=00:21:f6:00:00:00 to Network name=MyNetwork
```

See Also

- Section A.56, “create Vnic”
- Section A.95, “edit Vnic”
- Section A.142, “remove Vnic”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.18 changeServerAgentPassword

Changes the Oracle VM Agent password on an Oracle VM Server.

Syntax

changeServerAgentPassword Server *instance* oldPassword=*value* newPassword=*value* confirmPassword=*value*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command changes the Oracle VM Agent password on an Oracle VM Server.

Options

The following table shows the available options for this command.

Option	Description
`oldPassword=`*`value`*	The existing password for the Oracle VM Agent on the Oracle VM Server. The password is displayed as asterisks.
`newPassword=`*`value`*	The new password for the Oracle VM Agent on the Oracle VM Server. The password is displayed as asterisks.
`confirmPassword=`*`value`*	The new password for the Oracle VM Agent on the Oracle VM Server. The password is displayed as asterisks.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.25 Changing the Oracle VM Agent password on an Oracle VM Server

```
OVM> changeServerAgentPassword Server name=MyServer oldPassword=***** \
  newPassword=******* confirmPassword=*******
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.19 checkUpToDate

Checks whether the Oracle VM Server software is up-to-date according to the server update repository.

Syntax

`checkUpToDate` `Server` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command checks whether the Oracle VM Server software is up-to-date according to the server update repository. This command sets the `Up To Date` parameter of the Server object, and does not display any output other than a success or failure message. This command may be useful to check whether an Oracle VM Server is up-to-date in between any regular checking by the recurring job that checks for available updates. To see the value of the Server object's `Up To Date` parameter, use the `show Server` command. To update an Oracle VM Server, use the upgrade command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.26 Checking whether an Oracle VM Server is up-to-date

```
OVM> checkUpToDate Server name=MyServer
```

See Also

- Section A.46, “create ServerUpdateRepository”
- Section A.83, “edit ServerUpdateRepository”
- Section A.45, “create ServerUpdateGroup”
- Section A.82, “edit ServerUpdateGroup”
- Section A.19, “checkUpToDate”
- Section A.160, “upgrade”
- Section A.151, “show”

A.20 clearVmAllRcvdMessages

Clears all the key/value pair messages received by a running virtual machine.

Syntax

`clearVmAllRcvdMessages` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clears all the key/value pair messages received by a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.27 Clearing all messages received by a virtual machine

```
OVM> clearVmAllRcvdMessages Vm name=MyVm
```

See Also

- Section A.146, “sendVmMessage”
- Section A.112, “getVmSentMessages”

- Section A.111, “getVmReceivedMessages”
- Section A.22, “clearVmRcvdMessage”
- Section A.23, “clearVmSentMessage”
- Section A.21, “clearVmAllSentMessages”

A.21 clearVmAllSentMessages

Clears all the key/value pair messages sent to a running virtual machine.

Syntax

`clearVmAllSentMessages` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clears all the key/value pair messages sent to a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.28 Clearing all messages sent to a virtual machine

```
OVM> clearVmAllSentMessages Vm name=MyVm
```

See Also

- Section A.146, “sendVmMessage”
- Section A.112, “getVmSentMessages”
- Section A.111, “getVmReceivedMessages”
- Section A.22, “clearVmRcvdMessage”
- Section A.23, “clearVmSentMessage”
- Section A.20, “clearVmAllRcvdMessages”

A.22 clearVmRcvdMessage

Clears a key/value pair message received by a running virtual machine.

Syntax

`clearVmRcvdMessage Vm` *`instance`* `key=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clears a key/value pair message received by a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
`key=`*`value`*	The name or ID of the message key.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.29 Clearing a message received by a virtual machine

```
OVM> clearVmRcvdMessage Vm name=MyVm key="com.oracle.linux.network.device.0"
```

See Also

- Section A.146, “sendVmMessage”
- Section A.112, “getVmSentMessages”
- Section A.111, “getVmReceivedMessages”
- Section A.23, “clearVmSentMessage”
- Section A.21, “clearVmAllSentMessages”
- Section A.20, “clearVmAllRcvdMessages”

A.23 clearVmSentMessage

Clears a key/value pair message sent to a running virtual machine.

Syntax

`clearVmSentMessage Vm` *`instance`* `key=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clears a key/value pair message sent to a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
`key=`*`value`*	The name or ID of the message key.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.30 Clearing a message sent to a virtual machine

```
OVM> clearVmSentMessage Vm name=MyVm key="com.oracle.linux.network.device.0"
```

See Also

- Section A.146, “sendVmMessage”
- Section A.112, “getVmSentMessages”
- Section A.111, “getVmReceivedMessages”
- Section A.22, “clearVmRcvdMessage”
- Section A.21, “clearVmAllSentMessages”
- Section A.20, “clearVmAllRcvdMessages”

A.24 clone Vm

Clones a virtual machine or template to a new virtual machine or template.

Syntax

`clone Vm` *`instance`* `destType=` { `Vm` | `VmTemplate` } [`destName=`*`value`*] `serverPool=`*`value`* [`cloneCustomizer=`*`value`*] [`targetRepository=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a virtual machine or template to a new virtual machine or template.

Options

The following table shows the available options for this command.

Option	Description
`destType=` { `Vm` | `VmTemplate` }	The object to create from the virtual machine, either a virtual machine or a template.
`destName=`*`value`*	The name of the cloned virtual machine or template.

Option	Description
`serverPool=`*`value`*	The name or ID of the server pool on which to deploy the cloned virtual machine. **Note** Although you must enter this when cloning a virtual machine to a template, the template is not deployed to a server pool, it is located in the storage repository.
`cloneCustomizer=`*`value`*	The name or ID of the cloneCustomizer that should be used when deploying the cloned virtual machine or template.
`targetRepository=`*`value`*	The name or ID of the repository that should be used when deploying the cloned virtual machine or template.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.31 Cloning a virtual machine to a virtual machine

```
OVM> clone Vm name=MyVM destType=Vm destName=MyNewVM serverPool=MyServerPool
```

Example A.32 Cloning a virtual machine to a template

```
OVM> clone Vm name=MyVM destType=VmTemplate destName=MyVMTemplate serverPool=MyServerPool
```

Example A.33 Cloning a template to a virtual machine

```
OVM> clone Vm name=MyVMTemplate.tgz destType=Vm destName=MyNewVM serverPool=MyServerPool
```

Example A.34 Cloning a template to a template

```
OVM> clone Vm name=MyVMTemplate.tgz destType=VmTemplate destName=MyVMTemplate \
  serverPool=MyServerPool
```

Example A.35 Cloning a virtual machine to a virtual machine using a clone customizer

```
OVM> clone Vm name=MyVM destType=Vm destName=MyNewVM serverPool=MyServerPool \
    cloneCustomizer=MyCloneCustomizer targetRepository=MyRepository
```

See Also

- Section A.51, “create Vm”
- Section A.16, “add Vm”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.25 cloneCdToRepo

Clones a virtual CDROM to a storage repository.

Syntax

`cloneCdToRepo VirtualCdrom` *`instance`* `target=`*`value`* `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a virtual CDROM to a target repository.

Options

The following table shows the available options for this command.

Option	Description
`target=`*`value`*	The repository on which to locate the cloned physical disk.
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	Whether to clone a sparse, non-sparse or thin-clone virtual cdrom. The *`THIN_CLONE`* parameter is only applicable when using an Oracle VM Storage Connect plug-in that supports thin provisioned cloning. The clone target must be an OCFS2-based storage repository, or a storage array.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDisk1`.

Examples

Example A.36 Cloning a virtual CDROM to a repository

```
OVM> cloneCdToRepo VirtualCdrom name=MyCD.iso target=MyRepository cloneType=SPARSE_COPY
```

See Also

- Section A.117, “importVirtualCdrom”
- Section A.87, “edit VirtualCdrom”
- Section A.26, “clonePdToPd”
- Section A.28, “clonePdToStorageArray”
- Section A.39, “create PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

- Section A.103, “getEvents”

A.26 clonePdToPd

Clones a physical disk to a physical disk.

Syntax

`clonePdToPd PhysicalDisk` *`instance`* `target=`*`value`* `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a physical disk to a target physical disk. You cannot clone a physical disk using this command if the disk contains a file system or storage repository.

Options

The following table shows the available options for this command.

Option	Description
`target=`*`value`*	The physical disk on which to locate the cloned physical disk.
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	Whether to clone a sparse, non-sparse or thin-clone physical disk. The *`THIN_CLONE`* parameter is only applicable when using an Oracle VM Storage Connect plug-in that supports thin provisioned cloning. The clone target must be an OCFS2-based storage repository, or a storage array.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDisk1`.

Examples

Example A.37 Cloning a physical disk to a physical disk

```
OVM> clonePdToPd PhysicalDisk name=MyDisk1 target=MyRepository cloneType=SPARSE_COPY
```

See Also

- Section A.29, “cloneVdToPd”
- Section A.28, “clonePdToStorageArray”
- Section A.39, “create PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”

- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.27 clonePdToRepo

Clones a physical disk to a repository.

Syntax

`clonePdToRepo PhysicalDisk` *`instance`* `target=`*`value`* `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a physical disk to a target repository. The physical disk data is copied to a virtual disk image (`.img` file) in the storage repository.

Options

The following table shows the available options for this command.

Option	Description
`target=`*`value`*	The repository on which to locate the cloned physical disk. The target repository must be on a physical disk.
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	Whether to clone a sparse, non-sparse or thin-clone physical disk. The *`THIN_CLONE`* parameter is only applicable when using an Oracle VM Storage Connect plug-in that supports thin provisioned cloning. The clone target must be an OCFS2-based storage repository, or a storage array.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDisk1`.

Examples

Example A.38 Cloning a physical disk to a repository

```
OVM> clonePdToRepo PhysicalDisk name=MyDisk1 target=MyRepository cloneType=SPARSE_COPY
```

See Also

- Section A.26, “clonePdToPd”
- Section A.28, “clonePdToStorageArray”
- Section A.39, “create PhysicalDisk”

- Section A.74, “edit PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.28 clonePdToStorageArray

Clones a physical disk to a storage array.

Syntax

`clonePdToStorageArray PhysicalDisk` *`instance`* `target=`*`value`* `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` } `userFriendlyName=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a physical disk to a target storage array.

This command is not supported with a generic ISCSI Oracle VM Storage Connect plug-in. The clone target must be on the same storage array as the source. You cannot clone a disk from one storage array to another.

Options

The following table shows the available options for this command.

Option	Description
`target=`*`value`*	The storage array on which to locate the cloned physical disk.
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	Whether to clone a sparse, non-sparse or thin-clone physical disk.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDisk1`.
`userFriendlyName=`*`value`*	An optional parameter to specify a user-friendly name that can be used to identify the physical disk object.

Examples

Example A.39 Cloning a physical disk to a storage array

```
OVM> clonePdToStorageArray PhysicalDisk name=MyDisk1 target=MyRepository cloneType=SPARSE_COPY
```

See Also

- Section A.26, “clonePdToPd”

- Section A.28, “clonePdToStorageArray”
- Section A.39, “create PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.29 cloneVdToPd

Clones a virtual disk to a physical disk.

Syntax

`cloneVdToPd VirtualDisk` *`instance`* `target=`*`value`* `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a virtual disk to a target physical disk.

Options

The following table shows the available options for this command.

Option	**Description**
`target=`*`value`*	The physical disk on which to locate the cloned virtual disk.
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	Whether to clone a sparse, non-sparse or thin-clone virtual disk.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVMDisk`.

Examples

Example A.40 Cloning a virtual disk to a physical disk

```
OVM> cloneVdToPd VirtualDisk name=MyVMDisk target=MyDisk1 cloneType=SPARSE_COPY
```

See Also

- Section A.30, “cloneVdToRepo”
- Section A.118, “importVirtualDisk”

- Section A.49, “create VirtualDisk”
- Section A.88, “edit VirtualDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.30 cloneVdToRepo

Clones a virtual disk to a repository.

Syntax

`cloneVdToRepo VirtualDisk` *`instance`* `target=`*`value`* `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command clones a virtual disk to a target repository.

The virtual disk instance must be in a storage repository on a physical disk.

Options

The following table shows the available options for this command.

Option	Description
`target=`*`value`*	The repository on which to locate the cloned virtual disk. The target repository must be on a physical disk.
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	Whether to clone a sparse, non-sparse or thin-clone virtual disk. The *`THIN_CLONE`* parameter is only applicable when using an Oracle VM Storage Connect plug-in that supports thin provisioned cloning. The clone target must be an OCFS2-based storage repository, or a storage array.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVMDisk`.

Examples

Example A.41 Cloning a virtual disk to a repository

```
OVM> cloneVdToRepo VirtualDisk name=MyVirtualDisk target=MyRepository cloneType=SPARSE_COPY
```

See Also

- Section A.29, “cloneVdToPd”
- Section A.118, “importVirtualDisk”
- Section A.49, “create VirtualDisk”
- Section A.88, “edit VirtualDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.31 createVmFromAssembly

Creates a virtual machine template from an assembly file.

Syntax

`createVmFromAssembly` `AssemblyVm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a virtual machine template from an AssemblyVm object. An AssemblyVm object is created for each virtual machine in an assembly file when an assembly file is imported using the importAssembly command. The virtual machine template files are created in the same storage repository as the original AssemblyVm object. To create a virtual machine from the template, use the clone Vm command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAssembly.ova`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.42 Creating virtual machines from an assembly

```
OVM> createVmFromAssembly AssemblyVm name=myassembly.ova
```

See Also

- Section A.115, “importAssembly”
- Section A.24, “clone Vm”
- Section A.61, “edit Assembly”
- Section A.62, “edit AssemblyVirtualDisk”
- Section A.63, “edit AssemblyVm”
- Section A.102, “getDescriptor”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.32 create AccessGroup

Creates an access group.

Syntax

`create AccessGroup` `name=`*`value`* [`description=`*`value`*] `on` { `FileServer` | `StorageArray` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates an access group for either a file server or storage array. Generic storage array plug-ins are not supported with this command.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name for the access group.
`description=`*`value`*	Optional description for the access group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyFileServer`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.43 Creating an access group on a storage array

```
OVM> create AccessGroup name=MyAccessGroup on StorageArray name=MyISCSIServer
```

Example A.44 Creating an access group on a file system

```
OVM> create AccessGroup name=MyAccessGroup on FileServer name=MyNFSServer
```

See Also

- Section A.59, “edit AccessGroup”
- Section A.9, “add PhysicalDisk”
- Section A.134, “remove PhysicalDisk”
- Section A.37, “create FileSystem”
- Section A.70, “edit FileSystem”
- Section A.8, “add FileSystem”
- Section A.133, “remove FileSystem”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.121, “list”
- Section A.151, “show”
- Section A.57, “delete”
- Section A.103, “getEvents”

A.33 create AntiAffinityGroup

Creates an anti affinity group in a server pool.

Syntax

`create AntiAffinityGroup` `name=`*`value`* [`description=`*`value`*] `on ServerPool` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates an anti affinity group in a server pool. To add a virtual machine to an anti affinity group, use the add Vm command. To remove a virtual machine from an anti affinity group, use the remove Vm command.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the anti affinity group.
`description=`*`value`*	Optional description for the anti affinity group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.45 Creating an anti affinity group

```
OVM> create AntiAffinityGroup name=MyAAGroup on ServerPool name=MyServerPool
```

See Also

- Section A.60, “edit AntiAffinityGroup”
- Section A.16, “add Vm”
- Section A.141, “remove Vm”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.34 create BondPort

Creates a bond port on an Oracle VM Server.

Syntax

`create BondPort` `ethernetPorts=`*`value`* `mode=` { `ACTIVE_PASSIVE` | `LINK_AGGREGATION` | `LOAD_BALANCED` } `mtu=`*`value`* [`interfaceName=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on Server` *`instance`*

Where *`instance`* is:

{ id=*value* | name=*value* }

Description

This command creates a bond port on an *Oracle VM Server* .

To configure the IP address for a bond port, use the embeddedCreate command.

Options

The following table shows the available options for this command.

Option	Description
ethernetPorts=*value*	The name or ID of at least two Ethernet ports in a comma separated list. The name or ID must match the name or ID as it is stored for each port within Oracle VM Manager.
mode= { ACTIVE_PASSIVE | LINK_AGGREGATION | LOAD_BALANCED }	The network *bonding* mode.
mtu=*value*	The MTU value. May be an integer between 1500 and 64000.
interfaceName=*value*	An optional name for the bond in the format bond*N*, for example bond1, or bond2. If you do not enter a name, the default of bond*N* is used, where *N* is the next available bond number. This cannot be changed after the bond is created.
name=*value*	A name to identify the bond.
description=*value*	Optional description for the bond. *value* is a maximum of 4,000 characters.
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyServer.

Note

Any create command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.46 Creating a bond port

```
OVM> create BondPort mode=LINK_AGGREGATION mtu=1500 name=MyPortBond \
  ethernetPorts="0004fb0000200000d9394992f8ba06d4,0004fb0000200000c2a5f26641825be5" \
  on Server name=MyServer
```

```
OVM> create bondPort name=MyPortBond2 mode=ACTIVE_PASSIVE mtu=1500 \
  ethernetPorts="eth3 on MyServer,eth5 on MyServer" on server name=MyServer
```

See Also

- Section A.7, “add BondPort”
- Section A.64, “edit BondPort”

- Section A.132, “remove BondPort”
- Section A.135, “remove Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.35 create CpuCompatibilityGroup

Creates a *CPU compatibility group* to which *Oracle VM Server*s sharing a common processor can be assigned.

Syntax

`create CpuCompatibilityGroup` `name=`*`value`* [`description=`*`value`*]

Description

This command creates a CPU compatibility group to which Oracle VM Servers sharing a common processor can be assigned.

Options

The following table shows the available options for this command.

Option	**Description**
`name=`*`value`*	A name for the CPU compatibility group.
`description=`*`value`*	Optional description for the CPU compatibility group. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.47 Creating a CPU compatibility group

```
OVM> create CpuCompatibilityGroup name=MyCPUGroup
```

See Also

- Section A.67, “edit CpuCompatibilityGroup”

- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.121, “list”
- Section A.151, “show”
- Section A.57, “delete”
- Section A.103, “getEvents”

A.36 create FileServer

Discovers a file server.

Syntax

```
create FileServer plugin=value accessHost=value adminServers=value
[ refreshServers=value ] [ uniformedExports= { Yes | No } ] [ name=value ] [
description=value ]
```

Description

This command discovers a file server.

After discovering a file server, you should:

- Optionally, if you are using non-uniformed file system exports, you can create an access group using the create AccessGroup command. Add file systems to the access group using the add FileSystem command. Add Oracle VM Servers to the access group using the add Server command.
- Refresh the file server and file systems using the refresh command.

Options

The following table shows the available options for this command.

Option	Description
`plugin=value`	The name or ID of storage plug-in to use for the file server to be discovered. To obtain a list of existing plug-ins enter: `OVM> list FileServerPlugin` If a vendor specific plug-in is configured it is made available as an option here.
`accessHost=value`	The host name or IP address for the file server to be discovered.
`adminServers=value`	The names or IDs of the Oracle VM Servers to perform administration on the file server, in a comma separated list.
`refreshServers=value`	Optionally, the host names or IP addresses of the Oracle VM Servers to perform refresh jobs on the file server, in a comma separated list.

Option	Description
`uniformedExports=` { `Yes` | `No` }	Whether the file server has uniformed file system exports. The default is `Yes`.
`name=`*`value`*	A name to identify the file server.
`description=`*`value`*	Optional description for the file server. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.48 Discovering a file server

```
OVM> create FileServer plugin="Oracle Generic Network File System" \
  accessHost=10.172.76.125 adminServers="MyServer1.virtlab.info,MyServer2.virtlab.info" \
  name=MyNFSServer
```

See Also

- Section A.68, “edit FileServer”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.6, “addRefreshServer”
- Section A.131, “removeRefreshServer”
- Section A.124, “refresh”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.37 create FileSystem

Creates an OCFS2 file system on a physical disk on a file server.

Syntax

`create FileSystem` `physicalDisk=`*`value`* `name=`*`value`* [`description=`*`value`*] `on FileServer` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates an OCFS2 file system on a physical disk on a file server. When creating a file system on an NFS file server, you can use the file server itself to create the file system. When creating an OCFS2 file system on a LUN, you should use a local file server on an Oracle VM Server that has access to the LUN to create the file system. See Section 7.2, "To create a storage repository on a storage array" for an example.

Options

The following table shows the available options for this command.

Option	Description
`physicalDisk=`*`value`*	The name or ID of the physical disk on which to create the OCFS2 file system.
`name=`*`value`*	A name for the file system.
`description=`*`value`*	Optional description for the file system. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyFileSystem`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.49 Creating an OCFS2 file system on physical disk on a file server

```
OVM> create FileSystem physicalDisk=0004fb0000180000e3c93dc542901b7a name=MyRepoFileSystem \
 on FileServer id=0004fb00000900007e1ce0c83b3f136f
```

Example A.50 Creating an OCFS2 file system on a LUN

```
OVM> create FileSystem name=MyRepoFileSystem physicalDisk="MyLUN1" on FileServer \
  name="Local FS MyServer1"
```

See Also

- Section A.70, "edit FileSystem"
- Section A.8, "add FileSystem"
- Section A.133, "remove FileSystem"
- Section A.32, "create AccessGroup"

- Section A.59, “edit AccessGroup”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.121, “list”
- Section A.151, “show”
- Section A.57, “delete”
- Section A.103, “getEvents”

A.38 create Network

Creates an Ethernet-based network.

Syntax

`create Network` [`roles=` { `MANAGEMENT` | `LIVE_MIGRATE` | `CLUSTER_HEARTBEAT` | `VIRTUAL_MACHINE` | `STORAGE` }] `name=`*`value`* [`description=`*`value`*] [`on Server` *`instance`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates an Ethernet-based network. To create a local network on an Oracle VM Server, use the `on Server` *`instance`* option. You can only create a local network for virtual machine networks (using the `roles=VIRTUAL_MACHINE` option), as shown in Example A.52, “Creating a local network on an Oracle VM Server”. You cannot use a local network for traffic such as storage, or cluster heartbeat.

Options

The following table shows the available options for this command.

Option	**Description**
`roles=` { `MANAGEMENT` | `LIVE_MIGRATE` | `CLUSTER_HEARTBEAT` | `VIRTUAL_MACHINE` | `STORAGE` }	The network roles. Enter options separated by commas (,), for example: `roles='VIRTUAL_MACHINE,STORAGE'`
`name=`*`value`*	A name to identify the network.
`description=`*`value`*	Optional description for the network. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.51 Creating a network

```
OVM> create Network name=MyVMNetwork roles=VIRTUAL_MACHINE
```

Example A.52 Creating a local network on an Oracle VM Server

```
OVM> create Network name=MyLocalNetwork roles=VIRTUAL_MACHINE on Server name=MyServer
```

See Also

- Section A.10, “add Port”
- Section A.75, “edit Port”
- Section A.135, “remove Port”
- Section A.72, “edit Network”
- Section A.34, “create BondPort”
- Section A.121, “list”
- Section A.151, “show”

A.39 create PhysicalDisk

Creates a physical disk on a volume group.

Syntax

`create PhysicalDisk` `size=`*`value`* [`extraInfo=`*`value`*] `shareable=` { `Yes` | `No` } `thinProvision=` { `Yes` | `No` } `userFriendlyName=`*`value`* `name=`*`value`* [`description=`*`value`*] `on VolumeGroup` ***`instance`***

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a physical disk on a volume group. Local storage and generic storage plug-ins are not supported with this command.

Options

The following table shows the available options for this command.

Option	Description
`size=`*`value`*	The size of the physical disk in GiB. The value can be from `1` to `2048`.
`extraInfo=`*`value`*	
`shareable=` { `Yes` | `No` }	Whether the physical disk is shareable. Shareable disks have read/write privileges in multiple virtual machines and should be used with caution.

Option	Description
`thinProvision=` { `Yes` | `No` }	Whether to create a thin or non-thin physical disk.
`userFriendlyName=`*`value`*	A user friendly name to identify the disk.
`name=`*`value`*	A name to identify the physical disk.
`description=`*`value`*	Optional description for the physical disk. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVolumeGroup`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.53 Creating a physical disk on a volume group

```
OVM> create PhysicalDisk size=10 name=MyPhysicalDisk shareable=No \
  thinProvision=No userFriendlyName="My Disk" on VolumeGroup \
  id='Storage_Volume_Group @ 0004fb0000090000325a36dad3b3b7d8'
```

See Also

- Section A.9, “add PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.134, “remove PhysicalDisk”
- Section A.143, “resize”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.40 create Repository

Creates a storage repository.

Syntax

`create Repository` [`sharePath=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on FileSystem` *`instance`*

Where *`instance`* is:

{ `id=value` | `name=value` }

Note that if the instance name contains forward slashes and you need to quote the name, you must escape forward slashes by using additional forward slashes. This is illustrated in the example.

Description

This command creates a storage repository on a file system. To create a repository on a LUN, you should first create an OCFS2 file system on it using the create FileSystem command.

After you create a repository, you should refresh it.

Options

The following table shows the available options for this command.

Option	Description
`sharePath=value`	A path to a subdirectory on the selected file system.
`name=value`	A name to identify the storage repository.
`description=value`	Optional description for the storage repository. *value* is a maximum of 4,000 characters.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyFileSystem`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.54 Creating a storage repository on a file server

```
OVM> create Repository name=MyRepository on FileSystem \
  name="nfs on 10.172.76.125://mnt//vol2//repo01"
```

Example A.55 Creating a storage repository on a SAN server

```
OVM> create Repository name=MyRepository on FileSystem name=MyRepoFileSystem
```

See Also

- Section A.11, “add Server”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.41 create RepositoryExport

Creates a repository export.

Syntax

`create RepositoryExport clientHostName=`*`value`* `name=`*`value`* `repository=`*`value`* [`description=`*`value`*] `options=`*`value`* `on Server`*`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates an export on an Oracle VM Server to enable access for a third party back up tool to back up the contents of an OCFS2-based storage repository .

Options

The following table shows the available options for this command.

Option	Description
`clientHostName=`*`value`*	The hostname or IP address of the computer for which to grant access to the storage repository contents. This is likely to be the machine on which the third party back up and restore software is running.
`name=`*`value`*	A name to identify the export on the file server.
`repository=`*`value`*	An OCFS2-based storage repository presented to the Oracle VM Server. This is the repository to configure for back up.
`description=`*`value`*	Optional description for the export on the file server. *`value`* is a maximum of 4,000 characters.
`options=`*`value`*	The parameters to include in the NFS mount configuration, for example: `rw, async, no_root_squash, wdelay`. When no `options` are specified, the default NFS options on the Oracle VM Server are used.
`Server`*`instance`*	An Oracle VM Server on which the storage repository is presented.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=Server`.

Examples

Example A.56 Creating a repository export on an Oracle VM Server

```
OVM> create RepositoryExport clientHostName=10.172.76.146 name="My NFS Export" \
  repository=MyISCSIRepository options="rw, async, no_root_squash" \
  on Server name=MyServer1
```

See Also

- Section A.77, “edit RepositoryExport”
- Section A.57, “delete”

- Section A.121, “list”
- Section A.151, “show”

A.42 create ServerController

Creates a server controller object to configure IPMI on an Oracle VM Server.

Syntax

`create ServerController` `ipAddress=`*`value`* `userName=`*`value`* [`password=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on Server`*`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a server controller object to configure IPMI on an Oracle VM Server.

Options

The following table shows the available options for this command.

Option	**Description**
`virtualIP=`*`value`*	The IP address of the IPMI.
`userName=`*`value`*	The user name for the IPMI.
`password=`*`value`*	An optional password for the IPMI.
`name=`*`value`*	A name to identify the server control object.
`description=`*`value`*	Optional description for the server control object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerController`.

Examples

Example A.57 Configuring IPMI for an Oracle VM Server

```
OVM> create ServerController ipAddress=192.168.10.3 userName=admin password=password \
  name=MyServerController description="IPMI controller for MyServer1" on Server name=MyServer1
```

See Also

- Section A.79, “edit ServerController”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.43 create ServerPool

Creates a server pool.

Syntax

`create ServerPool virtualIP=`*`value`* `clusterEnable= {Yes|No}[clusterTimeout=`*`value`*`][fileSystem=`*`value`*`][physicalDisk=`*`value`*`][keymapName={en-us|ar|da|de|de-ch|en-gb|es|et|fi|fo|fr|fr-be|fr-ca|fr-ch|hr|hu|is|it|ja|lt|lv|mk|nl|nl-be|No|pl|pt|pt-br|ru|sl|sv|th|tr}][migrateUsingSsl= {Yes|No}][startPolicy= {BEST_SERVER|CURRENT_SERVER}][policyMode= {OFF|DRS|DPM}][policyCpuEnable= {Yes|No}][policyPeriod=`*`value`*`][policyCpuThreshold=`*`value`*`] name=`*`value`*`[description=`*`value`*`]`

Description

This command creates a server pool.

Options

The following table shows the available options for this command.

Option	**Description**
`virtualIP=`*`value`*	The virtual IP address for the server pool.
`clusterEnable= {Yes|No}`	Whether to enable a clustered server pool. If this parameter is not included, the default is `No`, so the server pool is not clustered. If this parameter is set to `Yes`, you must also include either the `fileSystem` or `physicalDisk` option to provide a location for the server pool file system.
`clusterTimeout=` *`value`*	Set the timeout value for the cluster in seconds. *`value`* is an integer between 30 and 300.
`fileSystem=`*`value`*	The file system to use for the server pool file system. Note that if you specify the name as the *`value`* for the file system, and the name is specified in quotes, any forward slashes in the name must be escaped using additional forward slashes. This is illustrated in the examples for this command.
`physicalDisk=`*`value`*	The physical disk to use for the server pool file system. **Note** You cannot create a server pool file system on a local physical disk as the server pool file system needs to be accessible by all Oracle VM Servers in the server pool.
`keymapName={en-us|ar|da|de|de-ch|en-gb|es|et|fi|fo|fr|fr-be|fr-ca|fr-ch|hr|hu|is|it|ja|lt|lv|mk|nl|nl-be|No|pl|pt|pt-br|ru|sl|sv|th|tr}`	The key mapping to be used when connecting to a virtual machine's console.
`migrateUsingSsl= {Yes|No}`	Whether to enable secure migration of virtual machines using SSL.

Option	Description
`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` }	The policy by which virtual machines are located when created in the server pool. If none is provided, the `CURRENT_SERVER` option is used by default.
`policyMode=` { `OFF` | `DRS` | `DPM` }	Set the policy to use for the server pool.
`policyCpuEnable=` { `Yes` | `No` }	Set whether to enable the policy set in the `policyMode` option for the server pool.
`policyPeriod=`*`value`*	The time period for the policy job to run. This sets the policy job to run every *n* minutes, for example, 10 sets the policy job to run every 10 minutes. *`value`* can be an integer between `2` and `60`.
`policyCpuThreshold=`*`value`*	The maximum amount of CPU percentage usage allowed before the policy must be enacted. *`value`* can be an integer between `0` and `99`.
`name=`*`value`*	A name to identify the server pool.
`description=`*`value`*	Optional description for the server pool. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.58 Creating a clustered server pool

```
OVM> create ServerPool virtualIP=10.172.77.195 clusterEnable=Yes \
  filesystem="nfs on 10.172.76.125://mnt//vol1//poolfs01" name=MyServerPool \
  description='Clustered server pool'
```

Example A.59 Creating an unclustered server pool

```
OVM> create ServerPool virtualIP=10.172.77.195 clusterEnable=No name=MyServerPool \
  description='Unclustered server pool'
```

See Also

- Section A.80, “edit ServerPool”
- Section A.44, “create ServerPoolNetworkPolicy”
- Section A.81, “edit ServerPoolNetworkPolicy”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.57, “delete”
- Section A.121, “list”

- Section A.151, “show”
- Section A.103, “getEvents”

A.44 create ServerPoolNetworkPolicy

Creates a server pool network policy.

Syntax

`create ServerPoolNetworkPolicy` [`policyEnable=` { `Yes` | `No` }] [`policyThreshold=`*`value`*] `network=`*`value`* `name=`*`value`* [`description=`*`value`*] `on` `ServerPool` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a server pool network policy. A server pool network policy is the object that controls DPM/DRS behavior of the virtual machines associated with the server pool based on network bandwidth usage.

Options

The following table shows the available options for this command.

Option	Description
`policyEnable=` { `Yes` | `No` }	Set whether to enable the policy set in the `policyMode` option of the create ServerPool or edit ServerPool command.
`policyThreshold=`*`value`*	The percentage (%) of network bandwidth the policy uses to move virtual machines. *`value`* can be an integer between 0 and 100.
`network=`*`value`*	The name or ID of the network associated with the policy, which is used to administer the policy for the server pool.
`name=`*`value`*	A name to identify the server pool network policy.
`description=`*`value`*	Optional description for the server pool network policy. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.60 Creating a server pool network policy

```
OVM> create ServerPoolNetworkPolicy network=MyNetwork policyEnable=Yes name=MyNetworkPolicy \
  on ServerPool name=MyServerPool
```

See Also

- Section A.81, “edit ServerPoolNetworkPolicy”
- Section A.5, “addPolicyServer”
- Section A.130, “removePolicyServer”
- Section A.43, “create ServerPool”
- Section A.80, “edit ServerPool”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.45 create ServerUpdateGroup

Creates an Oracle VM Server update group in a server pool.

Syntax

`create ServerUpdateGroup` `name=`*`value`* [`description=`*`value`*] on `ServerPool` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates an Oracle VM Server update group in a server pool. This allows you to override the default update group for a server pool. The default update groups:

- For x86-based server pools, have the ID of `GlobalX86ServerUpdateConfiguration`.
- For SPARC-based server pools, have the ID of `GlobalSparcServerUpdateConfiguration`.

When you have created a server update group for a server pool, you should then assign the update repository using the create ServerUpdateRepository command. To update an Oracle VM Server, use the upgrade command.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the server update group.
`description=`*`value`*	Optional description for the server update group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.61 Creating an update group for a server pool

```
OVM> create ServerUpdateGroup name=MyServerUpdateGroup on ServerPool name=MyServerPool
```

See Also

- Section A.82, “edit ServerUpdateGroup”
- Section A.46, “create ServerUpdateRepository”
- Section A.83, “edit ServerUpdateRepository”
- Section A.19, “checkUpToDate”
- Section A.160, “upgrade”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.46 create ServerUpdateRepository

Creates an Oracle VM Server update repository.

Syntax

`create ServerUpdateRepository` `repositoryName=`*`value`* `url=`*`value`* `enabled=` { `Yes` | `No` } `pkgSignatureType=` { `NONE` | `GPG` | `CA` } [`pkgSignatureKey=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on ServerUpdateGroup` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command sets the location for the YUM (for x86) or IPS (for SPARC) repository that contains the files to update Oracle VM Servers. This command creates an Oracle VM Server update repository and assigns it to a server update group. The default server update groups:

- For x86-based Oracle VM Servers, have the ID of `GlobalX86ServerUpdateConfiguration`.
- For SPARC-based Oracle VM Servers, have the ID of `GlobalSparcServerUpdateConfiguration`.

To override either of these default repositories and create a repository that is restricted to a particular server pool, you should first create a server update group for the server pool using the create ServerUpdateGroup command, then create an update repository and assign it to the new server update group. To update an Oracle VM Server, use the upgrade command.

Note that when a new server update repository is created, the repository is added to each of the servers that belong to the server update group. If there is a problem adding the repository to a server in the server update group, an error event is generated for that server within Oracle VM Manager. Oracle VM Manager does not attempt to validate the repository before it is added to each server. If the repository is invalid or, in the case of a SPARC repository, the repository name does not match a valid publisher at the URL specified, an error event is generated for the servers affected.

Options

The following table shows the available options for this command.

Option	Description
`repositoryName=`*`value`*	A name for the server update repository. This may only contain alphanumeric characters and underscores. No spaces are permitted. For SPARC repositories, this must match a valid publisher for the repository, hosted at the provided URL.
`url=`*`value`*	The URL to access the repository. If you enclose the URL in quotes, you must escape each forward slash (`/`) with another, for example: `url="http:////10.172.77.200//ovs"`
`enabled=` { `Yes` | `No` }	Whether to enable the repository.
`pkgSignatureType=` { `NONE` | `GPG` | `CA` }	The signature type to verify the validity of the repository, either `GPG` key (or GnuPG key), `CA` (Certificate Authority). Use `NONE` if there is no verification required.
`pkgSignatureKey=`*`value`*	The verification signature for the repository, for example, the location of the GPG key using any of the HTTP, FTP, FILE or HTTPS protocols. If you enclose the value for the option in quotes, you must escape each forward slash (`/`) with another, for example: `pkgSignatureKey="http:////10.172.77.200//ovs//RPM-GPG-KEY"`
`name=`*`value`*	A name to identify the server update repository.
`description=`*`value`*	Optional description for the server update repository. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.62 Creating an Oracle VM Server update repository

```
OVM> create ServerUpdateRepository repositoryName=MyUpdateRepository url=http://10.172.77.200/ovs \
   enabled=Yes pkgSignatureType=GPG pkgSignatureKey=http://10.172.77.200/ovs/RPM-GPG-KEY \
   name=Myx86Repository on ServerUpdateGroup id=GlobalX86ServerUpdateConfiguration
```

See Also

- Section A.83, “edit ServerUpdateRepository”
- Section A.45, “create ServerUpdateGroup”

- Section A.82, “edit ServerUpdateGroup”
- Section A.19, “checkUpToDate”
- Section A.160, “upgrade”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.47 create StorageArray

Discovers a storage array.

Syntax

`create StorageArray` `plugin=`*`value`* `storageType=` { `FIBRECHANNEL` | `ISCSI` | `LOCAL` | `UNKNOWN` } [`storageName=`*`value`*] [`accessHost=`*`value`*] [`accessPort=`*`value`*] [`accessUsername=`*`value`* `accessPassword=`*`value`*] [`useChap=` { `Yes` | `No` }] [`adminHost=`*`value`* `adminUserName=`*`value`* `adminPassword=`*`value`*] [`pluginPrivateData=`*`value`*] `name=`*`value`* [`description=`*`value`*]

Description

This command discovers a storage array and adds it to Oracle VM Manager. If you are adding a non-generic storage array also enter the additional plug-in options to enable Oracle VM Manager to access the storage array's configuration management functions using the `adminHost` option. To add more access hosts to enable *multipathing* on ISCSI servers, use the addAccessHost command.

After discovering a storage array, you should add storage initiators to it, add admin servers to it, validate it, then refresh it.

Options

The following table shows the available options for this command.

Option	Description
`plugin=`*`value`*	The name or ID of the Oracle VM Storage Connect plug-in to use for the storage array.
`storageType=` { `FIBRECHANNEL` | `ISCSI` | `LOCAL` | `UNKNOWN` }	The storage type for the storage array.
`storageName=`*`value`*	A name to identify the storage for a storage array behind a concentrator.
`accessHost=`*`value`*	The hostname or IP address for the storage array. This is not applicable to fibre channel storage arrays.
`accessPort=`*`value`*	The port on which access to the storage array is allowed. When adding iSCSI storage, add the access port as well. The default access port for iSCSI is `3260`. If not specified, the default port is used automatically.
`accessUsername=`*`value`*	A username with administrative access to the storage array, used with `accesshostname`. This option should only be used where CHAP is enabled on the storage array.

Option	Description
`accessPassword=`*`value`*	The password for the `accessusername` user. This option should only be used where CHAP is enabled on the storage array.
`useChap=` { `Yes` | `No` }	Whether to use CHAP authentication.
`adminHost=`*`value`*	The host name or IP address where administrative access to the storage array is allowed.
`adminUserName=`*`value`*	A user name with administrative access to the storage array, used with `adminHost`.
`adminPassword=`*`value`*	The administrator password for the `adminUserName` user.
`pluginPrivateData=`*`value`*	This option is used to pass additional parameters that a non-generic Oracle VM Storage Connect plug-in may accept to control functionality. For instance, in the case of an Oracle NetApp file system, you can enable SSL using this parameter in the following way: `pluginPrivateData="ssl=Yes"`
`name=`*`value`*	A name to identify the storage array.
`description=`*`value`*	Optional description for the storage array. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.63 Discovering a storage array

```
OVM> create StorageArray plugin="Oracle Generic SCSI Plugin" name=MyISCSIServer \
  storageType=ISCSI accessHost=10.172.76.130 accessPort=3260
```

See Also

- Section A.3, “addAccessHost”
- Section A.84, “edit StorageArray”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.124, “refresh”
- Section A.32, “create AccessGroup”
- Section A.13, “add StorageInitiator”

- Section A.161, “validate”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.48 create Tag

Creates a tag.

Syntax

`create Tag` `name=`*`value`* [`description=`*`value`*]

Description

This command creates a tag to identify and group objects.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the tag.
`description=`*`value`*	Optional description for the tag. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.64 Creating a tag

```
OVM> create Tag name=MyTag description="My tag."
```

See Also

- Section A.86, “edit Tag”
- Section A.14, “add Tag”
- Section A.139, “remove Tag”
- Section A.57, “delete”
- Section A.121, “list”

- Section A.151, “show”

A.49 create VirtualDisk

Creates a virtual disk.

Syntax

`create VirtualDisk` `size=`*`value`* `shareable=` { `Yes` | `No` } `sparse=` { `Yes` | `No` } `name=`*`value`* [`description=`*`value`*] `on Repository` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a virtual disk in a storage repository.

Options

The following table shows the available options for this command.

Option	**Description**
`size=`*`value`*	The size of the virtual disk in GiB. *`value`* can be an integer between `1` and `1000000`.
`shareable=` { `Yes` | `No` }	Whether the virtual disk is shareable. Shareable disks have read/write privileges in multiple virtual machines and should be used with caution.
`sparse=` { `Yes` | `No` }	Whether to create a sparse or non-sparse virtual disk.
`name=`*`value`*	A name to identify the virtual disk.
`description=`*`value`*	Optional description for the virtual disk. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyRepository`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.65 Creating a virtual disk in a storage repository

```
OVM> create VirtualDisk name=MyVMDisk size=10 sparse=Yes shareable=No on Repository \
  name=MyRepository
```

See Also

- Section A.118, “importVirtualDisk”

- Section A.88, “edit VirtualDisk”
- Section A.143, “resize”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.50 create VlanInterface

Creates a VLAN interface.

Syntax

`create VlanInterface` `vlanId=`*`value`* [`mtu=`*`value`*] `name=`*`value`* [`description=`*`value`*] `on` { `Port` | `BondPort` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a VLAN interface on either a port or a bond port.

To configure the IP address for a VLAN interface, use the embeddedCreate command.

Options

The following table shows the available options for this command.

Option	Description
`vlanId=`*`value`*	The ID of the VLAN interface. May be an integer between `2` and `4095`.
`mtu=`*`value`*	The MTU value. May be an integer between `1500` and `64000`.
`name=`*`value`*	A name to identify the VLAN interface.
`description=`*`value`*	Optional description for the VLAN interface. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyPort`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.66 Creating a VLAN interface

```
OVM> create VlanInterface vlanId=20 mtu=1500 name=MyVLANInterface on Port \
  id=0004fb0000200000229dbcccf17efec5
```

See Also

- Section A.89, “edit VlanInterface”
- Section A.15, “add VlanInterface”
- Section A.7, “add BondPort”
- Section A.64, “edit BondPort”
- Section A.132, “remove BondPort”
- Section A.135, “remove Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.51 create Vm

Creates a virtual machine.

Syntax

`create Vm` [`memory=`*`value`*] [`memoryLimit=`*`value`*] [`cpuCount=`*`value`*] [`cpuCountLimit=`*`value`*] [`cpuPriority=`*`value`*] [`cpuUtilizationCap=`*`value`*] [`highAvailability=` { `Yes` | `No` }] [`hugePagesEnabled=` { `Yes` | `No` }] [`osType=`*`value`*] [`mouseType=` { `OS_DEFAULT` | `PS2_MOUSE` | `USB_MOUSE` | `USB_TABLET` }] `domainType=` { `XEN_HVM` | `XEN_HVM_PV_DRIVERS` | `XEN_PVM` | `LDOMS_PVM` | `UNKNOWN` } [`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }] [`bootOrder=` { `PXE` | `DISK` | `CDROM` }] [`networkInstallPath=`*`value`*] `repository=`*`value`* [`server=`*`value`*] [`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` | `USE_POOL_POLICY` }] `name=`*`value`* [`description=`*`value`*] `on ServerPool` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a virtual machine.

Options

The following table shows the available options for this command.

Option	Description
`memory=`*`value`*	The memory size the virtual machine is allocated in MB. May be an integer between `32` and `2000000`.
`memoryLimit=`*`value`*	The maximum memory size the virtual machine can be allocated in MB. May be an integer between `32` and `2000000`.
`cpuCount=`*`value`*	The number of processors the virtual machine is allocated. May be an integer between `1` and `128`.
`cpuCountLimit=`*`value`*	The maximum number of processors the virtual machine can be allocated. May be an integer between `1` and `128`.
`cpuPriority=`*`value`*	The CPU priority of the virtual machine. A value between `1` and `100`; the higher the number, the more priority the CPU is given.
`cpuUtilizationCap=`*`value`*	The maximum percentage to which the virtual CPUs can receive scheduled time. A value between `10` and `100`; the higher the number, the more scheduled time the CPU is given.
`highAvailability=` { `Yes` | `No` }	Whether to enable High Availability (HA).
`hugePagesEnabled=` { `Yes` | `No` }	Whether to enable HugePages. If you enable HugePages, you should also enable support for HugePages in the guest operating system. This option must only be used for virtual machines that have their domain type set to XEN_PVM. More information on Huge Page support is provided in How are Huge Pages Enabled for Virtual Machines? in the *Oracle VM Concepts Guide*.
`osType=` *`value`*	The operating system of the virtual machine. To find the operating system type, use the getVmOsTypes command.
`mouseType=` { `OS_DEFAULT` | `PS2_MOUSE` | `USB_MOUSE` | `USB_TABLET` }	The mouse type of the virtual machine.
`domainType=` { `XEN_HVM` | `XEN_HVM_PV_DRIVERS` | `XEN_PVM` | `LDOMS_PVM` | `UNKNOWN` }	The domain type of the virtual machine.
`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }	The keyboard mapping to use for the virtual machine.
`bootOrder=` { `PXE` | `DISK` | `CDROM` }	The boot media order for the virtual machine. Enter options separated by commas (,), for example: `bootOrder='CDROM,DISK'` If you use the `PXE` boot option to boot from network-based installation media, also use the `networkInstallPath` parameter.
`networkInstallPath=`*`value`*	The location at which the installation media (mounted ISO file) is located when creating a PVM guest.

Option	Description
`repository=value`	The name or ID of the storage repository in which to create the virtual machine configuration file.
`server=value`	The name or ID of the Oracle VM Server on which to create the virtual machine.
`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` | `USE_POOL_POLICY` }	Optional virtual machine start up policy.
`name=value`	A name to identify the virtual machine.
`description=value`	Optional description for the virtual machine. *value* is a maximum of 4,000 characters.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.67 Creating a virtual machine

```
OVM> create Vm name=MyVM repository=MyRepository domainType=XEN_HVM \
  server=MyServer startPolicy=USE_POOL_POLICY on ServerPool name=MyServerPool
```

See Also

- Section A.110, “getVmOsTypes”
- Section A.24, “clone Vm”
- Section A.90, “edit Vm”
- Section A.16, “add Vm”
- Section A.156, “start”
- Section A.121, “list”
- Section A.151, “show”

A.52 create VmCloneCustomizer

Creates a clone customizer for a virtual machine.

Syntax

`create VmCloneCustomizer name=value` [`description=value`] `on Vm instance`

Where *instance* is:

{ `id=value` | `name=value` }

Description

This command allows you to create a new clone customizer for a virtual machine. When the clone customizer is created, you may want to create clone network and storage mappings that can be used by the clone customizer. See Section A.53, “create VmCloneNetworkMapping” and Section A.54, “create VmCloneStorageMapping” for more information.

Options

The following table shows the available options for this command.

Option	Description
`name=value`	A name to identify the clone customizer. **Tip** To find this name after the clone customizer is created, use the `list VmCloneCustomizer` command. You need the name or ID of this to delete a clone customizer from a virtual machine with the `delete VmCloneCustomizer` command.
`description=value`	Optional description for the clone customizer object. *value* is a maximum of 4,000 characters.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.68 Create a virtual machine clone customizer

```
OVM> create VmCloneCustomizer name=MyVmCloneCustomizer on Vm name=MyVM
```

See Also

- Section A.54, “create VmCloneStorageMapping”
- Section A.53, “create VmCloneNetworkMapping”
- Section A.91, “edit VmCloneCustomizer”
- Section A.57, “delete”
- Section A.121, “list”

- Section A.151, “show”

A.53 create VmCloneNetworkMapping

Maps the network that should be used by a virtual machine clone customizer.

Syntax

`create VmCloneNetworkMapping` `network=`*`value`* `vnic=`*`value`* `name=`*`value`* [`description=`*`value`*] `on vmCloneCustomizer` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

Creates a new network mapping for use by a virtual machine clone customizer. To create a clone network mapping, a virtual machine with an assigned VNIC must already exist within your environment.

Options

The following table shows the available options for this command.

Option	Description
`network=`*`value`*	The name or ID value of an existing network to which the cloned virtual machine should be connected.
`vnic=`*`value`*	The name or ID value of an existing VNIC that is attached to the existing virtual machine that is this clone customizer's parent.
`name=`*`value`*	A name to identify the clone network mapping. **Tip** To find this name after the network is mapped to a virtual machine clone customizer, use the `list vmCloneNetworkMapping` command. You need the name or ID of this to delete a network mapping from a virtual machine with the `delete vmCloneNetworkMapping` command.
`description=`*`value`*	Optional description for the clone network mapping object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVMCloneCustomizer`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.69 Creating a virtual machine clone network mapping

```
OVM> create VmCloneNetworkMapping name=MyCloneNet network=MyVMNetwork \
    vnic=0004fb0000070000277ecade9b897469 on VmCloneCustomizer name=MyVMCloneCustomizer
```

See Also

- Section A.52, “create VmCloneCustomizer”
- Section A.92, “edit VmCloneNetworkMapping”
- Section A.54, “create VmCloneStorageMapping”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.54 create VmCloneStorageMapping

Maps the storage for a virtual machine disk that should be used by a virtual machine clone customizer.

Syntax

`create VmCloneStorageMapping` `cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` } `vmDiskMapping=`*`value`* { `physicalDisk=`*`value`* | `repository=`*`value`* | `storageArray=`*`value`* } `name=`*`value`* [`description=`*`value`*] `on vmCloneCustomizer` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

Maps the storage for a virtual machine disk that should be used by a virtual machine clone customizer. A disk mapping that is already used by virtual machine that is this clone customizer's parent must already exist.

Options

The following table shows the available options for this command.

Option	Description
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	The type of *clone* storage that should be created.
`vmDiskMapping=`*`value`*	The name or ID of an existing virtual machine disk mapping.
{ `physicalDisk=`*`value`* | `repository=`*`value`* | `storageArray=`*`value`* }	The name or ID of either a physicalDisk, repository or storageArray object where the clone disk is to be created. You must specify at least one of these options and its associated value.
`name=`*`value`*	A name to identify the clone storage mapping.

Option	Description
	Tip To find this name after virtual storage is mapped to a virtual machine clone customizer, use the `list vmCloneStorageMapping` command. You need the name or ID of this to delete a disk mapping from a virtual machine clone customizer with the `delete vmCloneStorageMapping` command.
`description=`*`value`*	Optional description for the clone storage mapping object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVMCloneCustomizer`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.70 Creating a virtual machine clone storage mapping

```
OVM> create VmCloneStorageMapping name=BootDisk cloneType=SPARSE_COPY \
    vmDiskMapping=0004fb000013000096e5d46c5f5e6a52 repository=MyRepository \
    on VmCloneCustomizer name=MyVMCloneCustomizer
```

See Also

- Section A.52, “create VmCloneCustomizer”
- Section A.93, “edit VmCloneStorageMapping”
- Section A.53, “create VmCloneNetworkMapping”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.55 create VmDiskMapping

Maps a virtual disk, physical disk, or CDROM to a virtual machine disk slot.

Syntax

`create VmDiskMapping` `slot=`*`value`* { `physicalDisk=`*`value`* | `virtualDisk=`*`value`* | `virtualCd=` { *`value`* | `EMPTY_CDROM` } } `name=`*`value`* [`description=`*`value`*] `on Vm` *`instance`*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command maps a virtual disk, physical disk, or CDROM to a virtual machine disk slot. To create an empty CDROM drive, use the `virtualCd=EMPTY_CDROM` option. To edit a virtual disk or eject a CDROM, remove it using the `delete VmDiskMapping` command, then use the `create VmDiskMapping` command again to remap it to a virtual machine with any changed settings.

Options

The following table shows the available options for this command.

Option	Description
slot=*value*	The slot number for the disk in the virtual machine This can be an integer between `0` and `1000000`.
physicalDisk=*value*	The name or ID of the physical disk.
VirtualDisk=*value*	The name or ID of the virtual disk.
virtualCd= { *value* | EMPTY_CDROM }	The name or ID of the ISO file (virtual CDROM). Alternatively, to create an empty CDROM, use the `EMPTY_CDROM` option.
name=*value*	A name to identify the disk mapping. **Tip** To find this name after a virtual disk is mapped to a virtual machine, use the `list VmDiskMapping` command. You will need the name or ID of this to delete a disk mapping from a virtual machine with the `delete VmDiskMapping` command.
description=*value*	Optional description for the disk mapping object. *value* is a maximum of 4,000 characters.
{ id=*value* | name=*value* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.71 Mapping a virtual disk to a virtual machine

```
OVM> create VmDiskMapping slot=0 virtualDisk=MyVMDisk name="Boot Disk" on Vm name=MyVM
```

Example A.72 Mapping an ISO file (CDROM) to a virtual machine

```
OVM> create VmDiskMapping slot=1 virtualCd=OracleLinux-dvd.iso \
  name="CDROM Drive" on Vm name=MyVM
```

Example A.73 Mapping an empty CDROM drive to a virtual machine

```
OVM> create VmDiskMapping slot=2 virtualCd=EMPTY_CDROM name="CDROM Drive" on Vm name=MyVM
```

Example A.74 Mapping a physical disk to a virtual machine

```
OVM> create VmDiskMapping slot=3 physicalDisk=MyPhysicalDisk name="D Drive" on Vm name=MyVM
```

See Also

- Section A.94, “edit VmDiskMapping”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.56 create Vnic

Creates a VNIC on a virtual machine.

Syntax

`create Vnic` [`macAddress=`*`value`*] `network=`*`value`* `name=`*`value`* [`description=`*`value`*] `on Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command creates a VNIC and associates it to a virtual machine.

Options

The following table shows the available options for this command.

Option	Description
`network=`*`value`*	The name or ID of the network to which the VNIC is to be assigned.
`macAddress=`*`value`*	Optional parameter to specify the MAC address for the VNIC. If no MAC address is provided, one is automatically assigned from the default range. To change the default range of MAC addresses, use the setVnicMacAddrRange command. To display the MAC address range, use the getVnicMacAddrRange command.
`description=`*`value`*	Optional description for the VNIC. *`value`* is a maximum of 4,000 characters.

Note

Any `create` command only creates a single instance of an object, and therefore only accepts a single object instance as an attribute. Providing more than one object of the same attribute type as a parameter always results in the last attribute value taking precedence.

Examples

Example A.75 Creating a VNIC

```
OVM> create Vnic name=00:21:f6:00:00:18 network=MyVMNetwork on Vm name=MyVM
```

See Also

- Section A.95, “edit Vnic”
- Section A.17, “add Vnic”
- Section A.142, “remove Vnic”
- Section A.150, “setVnicMacAddrRange”
- Section A.113, “getVnicMacAddrRange”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.57 delete

Deletes an object.

Syntax

`delete` { `AccessGroup` | `AntiAffinityGroup` | `Assembly` | `BondPort` | `CpuCompatibilityGroup` | `FileServer` | `FileSystem` | `Network` | `PhysicalDisk` | `Repository` | `RepositoryExport` | `Server` | `ServerController` | `ServerPool` | `ServerPoolNetworkPolicy` | `ServerUpdateGroup` | `ServerUpdateRepository` | `StorageArray` | `Tag` | `VirtualCdrom` | `VirtualDisk` | `VlanInterface` | `Vm` | `VmCloneCustomizer` | `VmCloneNetworkMapping` | `VmCloneStorageMapping` | `VmDiskMapping` | `Vnic` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command deletes an object.

Virtual machines and virtual machine templates are treated as equivalent within the CLI. Therefore, to delete a virtual machine template you should use the `delete Vm` command. Since it is possible that a virtual machine template and a virtual machine may share the same name, it is recommended that you use the object's unique ID to perform this operation.

Options

The following table shows the available options for this command.

Option	Description
{ AccessGroup | AntiAffinityGroup | Assembly | BondPort | CpuCompatibilityGroup | FileServer | FileSystem | Network | PhysicalDisk | Repository | RepositoryExport | Server | ServerController | ServerPool | ServerPoolNetworkPolicy | ServerUpdateGroup | ServerUpdateRepository | StorageArray | Tag | VirtualCdrom | VirtualDisk | VlanInterface | Vm | VmCloneCustomizer | VmCloneNetworkMapping | VmCloneStorageMapping | VmDiskMapping | Vnic }	The object to delete. Before you delete a virtual machine (Vm object), you should make sure all disks have been removed from it using the `delete VmDiskMapping` command.
{ id=*value* | name=*value* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.76 Deleting an Oracle VM Server

```
OVM> delete Server name=MyServer
```

Example A.77 Deleting a virtual machine or virtual machine template

```
OVM> delete Vm id=0004fb00001400007be890778aedc7b8
```

Example A.78 Deleting a network

```
OVM> delete Network name=MyVMNetwork
```

Example A.79 Deleting a virtual machine disk mapping

```
OVM> delete VmDiskMapping id=0004fb00001300009d46acbb77de919e
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.58 discoverServer

Discovers an *Oracle VM Server*.

Syntax

discoverServer ipAddress=*value* password=*value* takeOwnership= { Yes | No }

Description

This command discovers an Oracle VM Server and adds it to Oracle VM Manager. The username used to connect to the Oracle VM Agent on the Oracle VM Server is `oracle`. The port number on which access is made to the Oracle VM Agent on the Oracle VM Server is `8899`.

Options

The following table shows the available options for this command.

Option	Description
`ipAddress=`*`value`*	The IP address or hostname of the Oracle VM Server.
`password=`*`value`*	The password to use when connecting to the Oracle VM Agent on the Oracle VM Server.
`takeOwnership=` { `Yes` | `No` }	Whether to take ownership of the Oracle VM Server. ⚠ **Important** The `takeOwnership` option should only be used if the Oracle VM Server is not already owned by an existing Oracle VM Manager. If specified for an Oracle VM Server that is already owned, the option is silently ignored.

Examples

Example A.80 Discovering an Oracle VM Server

```
OVM> discoverServer ipAddress=10.172.76.73 password=password takeOwnership=Yes
```

See Also

- Section A.78, “edit Server”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.156, “start”
- Section A.144, “restart”
- Section A.157, “stop”
- Section A.120, “kill”
- Section A.160, “upgrade”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

- Section A.103, “getEvents”

A.59 edit AccessGroup

Edits an access group.

Syntax

`edit AccessGroup` *`instance`* [`name=`*`value`*] [`nameOnArray=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an access group. Generic Oracle VM Storage Connect plug-ins are not supported with this command.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`name=`*`value`*	A name for the access group.
`nameOnArray=`*`value`*	A name for the access group as known on the storage array.
`description=`*`value`*	Optional description for the access group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAccessGroup`.

Examples

Example A.81 Changing the name of an access group

```
OVM> edit AccessGroup name=MyAccessGroup nameOnArray=MyNewName
```

See Also

- Section A.32, “create AccessGroup”
- Section A.9, “add PhysicalDisk”
- Section A.134, “remove PhysicalDisk”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.60 edit AntiAffinityGroup

Edits an anti affinity group in a server pool.

Syntax

`edit AntiAffinityGroup` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an anti affinity group in a server pool. To add a virtual machine to an anti affinity group, use the add Vm command. To remove a virtual machine from an anti affinity group, use the remove Vm command.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the anti affinity group.
`description=`*`value`*	Optional description for the anti affinity group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAAGroup`.

Examples

Example A.82 Editing an anti affinity group

```
OVM> edit AntiAffinityGroup name=MyAAGroup description="My Anti Affinity Group"
```

See Also

- Section A.33, “create AntiAffinityGroup”
- Section A.16, “add Vm”
- Section A.141, “remove Vm”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.61 edit Assembly

Edits an *assembly*.

Syntax

`edit Assembly` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of assembly.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the assembly.
`description=`*`value`*	Optional description for the assembly. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAssembly.ova`.

Examples

Example A.83 Editing an assembly

```
OVM> edit Assembly name=myassembly.ova description='Oracle Linux Release 6'
```

See Also

- Section A.62, “edit AssemblyVirtualDisk”
- Section A.63, “edit AssemblyVm”
- Section A.115, “importAssembly”
- Section A.102, “getDescriptor”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.62 edit AssemblyVirtualDisk

Edits a virtual disk in an *assembly*.

Syntax

`edit AssemblyVirtualDisk` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a virtual disk in an assembly. You cannot delete an AssemblyVirtualDisk object. You can only delete it by deleting the assembly itself.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the assembly virtual disk.
`description=`*`value`*	Optional description for the assembly virtual disk. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyAssemblyVirtualDisk`.

Examples

Example A.84 Editing an assembly virtual disk

```
OVM> edit AssemblyVirtualDisk name=MyAssemblyVirtualDisk description='Oracle Linux Release 6'
```

See Also

- Section A.61, “edit Assembly”
- Section A.63, “edit AssemblyVm”
- Section A.115, “importAssembly”
- Section A.102, “getDescriptor”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.63 edit AssemblyVm

Edits a virtual machine in an assembly.

Syntax

`edit AssemblyVm` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command edits the attributes of a virtual machine in an assembly. You cannot delete an AssemblyVm object. You can only delete it by deleting the assembly itself.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
name=*value*	A name to identify the assembly virtual machine.
description=*value*	Optional description for the assembly virtual machine. *value* is a maximum of 4,000 characters.
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyAssemblyVM.

Examples

Example A.85 Editing an assembly virtual machine

```
OVM> edit AssemblyVm name=MyAssemblyVM description='Oracle Linux Release 6'
```

See Also

- Section A.61, “edit Assembly”
- Section A.62, “edit AssemblyVirtualDisk”
- Section A.115, “importAssembly”
- Section A.102, “getDescriptor”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.64 edit BondPort

Edits a bond port.

Syntax

edit BondPort *instance* [mode= { ACTIVE_PASSIVE | LINK_AGGREGATION | LOAD_BALANCED }] [mtu=*value*] [name=*value*] [description=*value*]

Where *instance* is:

{ `id=value` | `name=value` }

Description

This command edits the attributes of a bond port on an Oracle VM Server.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`mode=` { `ACTIVE_PASSIVE` | `LINK_AGGREGATION` | `LOAD_BALANCED` }	The network bonding mode.
`mtu=value`	The MTU value. May be an integer between `1500` and `64000`.
`name=value`	A name to identify the bond.
`description=value`	Optional description for the bond. *`value`* is a maximum of 4,000 characters.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyPortBond`.

Examples

Example A.86 Editing a bond port

```
OVM> edit BondPort id=0004fb0000200000884da42c23947622 mode=LOAD_BALANCED
```

See Also

- Section A.34, “create BondPort”
- Section A.7, “add BondPort”
- Section A.132, “remove BondPort”
- Section A.135, “remove Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.65 edit ControlDomain

Edits a control domain.

Syntax

`edit ControlDomain` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a control domain. You cannot create or delete a control domain; you can only edit the name and description.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the control domain.
`description=`*`value`*	Optional description for the control domain. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyControlDomain`.

Examples

Example A.87 Editing a control domain

```
OVM> edit ControlDomain id=0004fb0000210000308f567ae2cbdc4c description="SPARC control domain"
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.66 edit Cpu

Edits a control domain.

Syntax

`edit Cpu` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a CPU. You cannot create or delete a CPU; you can only edit the name and description.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the CPU.
`description=`*`value`*	Optional description for the CPU. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyCPU`.

Examples

Example A.88 Editing a CPU

```
OVM> edit Cpu id="Processor (1) in 00:e0:81:4d:40:c6:00:e0:81:4d:40:c7:ff:ff:ff:ff" \
  description="CPU 1 on MyServer1"
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.67 edit CpuCompatibilityGroup

Edits a *CPU compatibility group*.

Syntax

`edit CpuCompatibilityGroup` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a CPU compatibility group.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the CPU compatibility group.
`description=`*`value`*	Optional description for the CPU compatibility group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyCPUGroup`.

Examples

Example A.89 Editing a CPU compatibility group

```
OVM> edit CpuCompatibilityGroup name=MyCPUGroup description='SPARC-based CPU compatibility group'
```

See Also

- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.35, “create CpuCompatibilityGroup”
- Section A.121, “list”
- Section A.151, “show”
- Section A.57, “delete”

A.68 edit FileServer

Edits a file server.

Syntax

`edit FileServer` *`instance`* [`plugin=`*`value`*] [`accessHost=`*`value`*] [`uniformedExports=` { `Yes` | `No` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a file server.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`plugin=`*`value`*	The name or ID for the Oracle VM Storage Connect plug-in for the file server. To see a list of the available plug-ins, use the `list StorageArrayPlugin` command.
`accessHost=`*`value`*	The host name or IP address for the file server.
`uniformedExports=` { `Yes` | `No` }	Whether the file server has uniformed file system exports. The default is `Yes`.
`name=`*`value`*	A name to identify the file server.
`description=`*`value`*	Optional description for the file server. *`value`* is a maximum of 4,000 characters.

Option	Description
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyNFSServer`.

Examples

Example A.90 Editing a file server

```
OVM> edit FileServer id=0004fb00000900000ef55b2f96a564c8 name=MyNFSServer \
  description='My NFS Server'
```

See Also

- Section A.36, “create FileServer”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.6, “addRefreshServer”
- Section A.131, “removeRefreshServer”
- Section A.124, “refresh”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.69 edit FileServerPlugin

Edits a file server plug-in.

Syntax

`edit FileServerPlugin` *`instance`* [`name=value`] [`description=value`]

Where *`instance`* is:

{ `id=value` | `name=value` }

Description

This command edits the attributes of a file server plug-in. You cannot create or delete a file server plug-in; you can only edit the name and description.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=value`	A name to identify the file server plug-in.
`description=value`	Optional description for the file server plug-in. *`value`* is a maximum of 4,000 characters.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyFileServerPlugin`.

Examples

Example A.91 Editing a file server plug-in

```
OVM> edit FileServerPlugin id="oracle.generic.NFSPlugin.GenericNFSPlugin (1.1.0)" \
  description="File Server Plugin for NFS Server"
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.70 edit FileSystem

Edits an OCFS2 file system.

Syntax

`edit FileSystem` *`instance`* [`name=value`] [`description=value`]

Where *`instance`* is:

{ `id=value` | `name=value` }

Description

This command edits the attributes of an OCFS2 file system.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=value`	A name to identify the file system.
`description=value`	Optional description for the file system. *`value`* is a maximum of 4,000 characters.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyFileSystem`.

Examples

Example A.92 Editing a file system

```
OVM> edit FileSystem id=0004fb0000050000002618dec56ee0e8 name=MyFileSystem \
  description='My File System'
```

See Also

- Section A.37, “create FileSystem”
- Section A.8, “add FileSystem”
- Section A.133, “remove FileSystem”
- Section A.32, “create AccessGroup”
- Section A.59, “edit AccessGroup”
- Section A.124, “refresh”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.71 edit Manager

Edits a Manager object.

Syntax

`edit Manager` *`instance`* `name=`*`value`* [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a Manager (Oracle VM Manager) object.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the Manager object.
`description=`*`value`*	Optional description for the Manager object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name="OVM Manager"`.

Examples

Example A.93 Renaming a Manager object

```
OVM> edit Manager name="OVM Manager" name="Oracle VM Manager"
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.72 edit Network

Edits an Ethernet-based network.

Syntax

`edit Network` *`instance`* [`roles=` { `MANAGEMENT` | `LIVE_MIGRATE` | `CLUSTER_HEARTBEAT` | `VIRTUAL_MACHINE` | `STORAGE` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an Ethernet-based network.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`roles=` { `MANAGEMENT` | `LIVE_MIGRATE` | `CLUSTER_HEARTBEAT` | `VIRTUAL_MACHINE` | `STORAGE` }	The network roles. Enter options separated by commas (,), for example: `roles='VIRTUAL_MACHINE,STORAGE'`
`name=`*`value`*	A name to identify the network.
`description=`*`value`*	Optional description for the network. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyNetwork`.

Examples

Example A.94 Editing a network

```
OVM> edit Network name=MyVMNetwork roles='VIRTUAL_MACHINE,LIVE_MIGRATION'
```

See Also

- Section A.10, “add Port”

- Section A.75, “edit Port”
- Section A.135, “remove Port”
- Section A.38, “create Network”
- Section A.34, “create BondPort”
- Section A.121, “list”
- Section A.151, “show”

A.73 edit PeriodicTask

Edits a periodic task.

Syntax

`edit PeriodicTask` *`instance`* [`taskInterval=`*`value`*] [`enabled=` { `Yes` | `No` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a periodic task. A periodic task is a task (job) that can be run multiple times, with an interval in between, such as checking server updates repositories for available Oracle VM Server software updates.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`taskInterval=`*`value`*	The interval between which the periodic task should run, in minutes.
`enabled=` { `Yes` | `No` }	Whether to enable the periodic task.
`name=`*`value`*	A name to identify the periodic task.
`description=`*`value`*	Optional description for the periodic task. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name="Server Upgrade Checker Task"`.

Examples

Example A.95 Disabling a periodic task

```
OVM> edit PeriodicTask name="Server Upgrade Checker Task" enabled=No
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.74 edit PhysicalDisk

Edits a physical disk.

Syntax

`edit PhysicalDisk` *`instance`* `shareable=` { `Yes` | `No` } [`extraInfo=`*`value`*] `name=`*`value`* [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a physical disk. *Local storage* and generic storage plug-ins are not supported with this command. To resize a physical disk, use the resize command.

Options

The following table shows the available options for this command.

Option	Description
`shareable=` { `Yes` | `No` }	Whether the physical disk is shareable. Shareable disks have read/write privileges in multiple virtual machines and should be used with caution.
`extraInfo=`*`value`*	Any extra parameters for your Oracle VM Storage Connect plug-in.
`name=`*`value`*	A name to identify the physical disk.
`description=`*`value`*	Optional description for the physical disk. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDisk`.

Examples

Example A.96 Editing a physical disk

```
OVM> edit PhysicalDisk id=0004fb000018000034a2da375d08990e shareable=Yes
```

See Also

- Section A.39, “create PhysicalDisk”
- Section A.9, “add PhysicalDisk”

- Section A.134, “remove PhysicalDisk”
- Section A.143, “resize”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.75 edit Port

Edits an Ethernet port on an Oracle VM Server.

Syntax

`edit Port` *`instance`* [`mtu=`*`value`*] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an Ethernet port on an Oracle VM Server .

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`mtu=`*`value`*	The MTU value. May be an integer between `1500` and `64000`.
`name=`*`value`*	A name to identify the port.
`description=`*`value`*	Optional description for the port. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `id=0004fb0000200000c2a5f26641825be5`.

Examples

Example A.97 Editing an Ethernet port

```
OVM> edit Port id=0004fb0000200000b0f9d86788b94a0e mtu=1500
```

See Also

- Section A.10, “add Port”

- Section A.34, “create BondPort”
- Section A.7, “add BondPort”
- Section A.132, “remove BondPort”
- Section A.135, “remove Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.76 edit Repository

Edits a storage repository.

Syntax

`edit Repository` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a storage repository.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`name=`*`value`*	A name to identify the storage repository.
`description=`*`value`*	Optional description for the storage repository. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyRepository`.

Examples

Example A.98 Editing a storage repository

```
OVM> edit Repository id=0004fb00000300003ab65ab35e3fea7a name=MyRepository \
  description="My Storage Repository"
```

See Also

- Section A.40, “create Repository”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.77 edit RepositoryExport

Edits a repository export.

Syntax

`edit RepositoryExport` *`instance`* `name=`*`value`* [`description=`*`value`*] [`options=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an export on an Oracle VM Server to enable access for a third party back up tool to back up the contents of an OCFS2-based storage repository .

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the export on the file server.
`description=`*`value`*	Optional description for the export on the file server. *`value`* is a maximum of 4,000 characters.
`options=`*`value`*	The parameters to include in the NFS mount configuration, for example: `rw, async, no_root_squash, wdelay`.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.99 Editing a repository export on an Oracle VM Server

```
OVM> edit RepositoryExport name="My NFS Export" name=MyExport \
  options="rw, async, no_root_squash"
```

See Also

- Section A.41, “create RepositoryExport”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.78 edit Server

Edits an Oracle VM Server.

Syntax

`edit Server` *`instance`* [`ntpServers=`*`value`*] [`roles=` { `VM` | `UTILITY` }] [`inboundMigrationLocked=` { `Yes` | `No` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an Oracle VM Server. Use this command set the NTP servers, role, and whether to allow running of virtual machines.

To place an Oracle VM Server into or out of maintenance mode,use the setMaintenanceMode command.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`ntpServers=`*`value`*	The hostname or IP address of one or more NTP servers to set the time on the Oracle VM Server. Enter options separated by commas (,), for example: `ntpServers="ntp1.example.com,ntp2.example.com"`
`roles=` { `VM` | `UTILITY` }	The role for the Oracle VM Server.
`inboundMigrationLocked=` { `Yes` | `No` }	Whether to allow additional virtual machines to run on the Oracle VM Server. Setting this option to *`Yes`* prevents new or migrated virtual machines from running on the Oracle VM Server. Setting this option to *`No`* allows new or migrated virtual machines to run on the Oracle VM Server. This option does not prevent virtual machines running on the Oracle VM Server from being migrated to another.
`name=`*`value`*	A name to identify the Oracle VM Server.
`description=`*`value`*	Optional description for the Oracle VM Server . *`value`* is a maximum of 4,000 characters.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.100 Adding NTP servers to an Oracle VM Server

```
OVM> edit Server name=MyServer ntpServer="ntp1.example.com,ntp2.example.com"
```

Example A.101 Changing the role of an Oracle VM Server

```
OVM> edit Server name=MyServer role=UTILITY runVmsEnabled=No
```

See Also

- Section A.58, “discoverServer”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.156, “start”
- Section A.144, “restart”
- Section A.157, “stop”
- Section A.120, “kill”
- Section A.148, “setMaintenanceMode”
- Section A.160, “upgrade”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.6, “addRefreshServer”
- Section A.131, “removeRefreshServer”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.79 edit ServerController

Edits a server controller object to configure IPMI on an Oracle VM Server.

Syntax

`edit ServerController` *`instance`* `ipAddress=`*`value`* `userName=`*`value`* [`password=`*`value`*] `name=`*`value`* [`description=`*`value`*]

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command edits the attributes of a server controller object to configure IPMI on an Oracle VM Server.

Options

The following table shows the available options for this command.

Option	Description
virtualIP=*value*	The IP address of the IPMI.
userName=*value*	The user name for the IPMI.
password=*value*	An optional password for the IPMI.
name=*value*	A name to identify the server control object.
description=*value*	Optional description for the server control object. *value* is a maximum of 4,000 characters.
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyServerController.

Examples

Example A.102 Editing IPMI configuration for an Oracle VM Server

```
OVM> edit ServerController id=mgtCtl_00:e0:81:4d:40:c6:00:e0:81:4d:40:c7:ff:ff:ff:ff \
  ipAddress=192.168.10.4 userName=admin password=password name=MyServerController
```

See Also

- Section A.42, “create ServerController”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.80 edit ServerPool

Edits a server pool.

Syntax

edit ServerPool *instance* clusterEnable= { Yes | No } [clusterTimeout= *value*] [keymapName= { en-us | ar | da | de | de-ch | en-gb | es | et | fi | fo | fr | fr-be | fr-ca | fr-ch | hr | hu | is | it | ja | lt | lv | mk | nl | nl-be | No | pl | pt | pt-br | ru | sl | sv | th | tr }] [migrateUsingSsl= { Yes | No }] [masterServer=*value*] [startPolicy= { BEST_SERVER | CURRENT_SERVER }] [policyMode= { OFF | DRS | DPM }] [policyCpuEnable= { Yes | No }] [policyPeriod=*value*] [policyCpuThreshold=*value*] [name=*value*] [description=*value*]

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command edits the attributes of a server pool. Use this command to change the master Oracle VM Server, change the keyboard mapping, set whether to use secure migration of virtual machines, and to manage server pool policies.

Although none of the options are mandatory, you must supply at least one option.

It is not possible to edit the cluster enable flag for a server pool if there are Oracle VM Servers in the server pool. Attempts to edit this attribute for a server pool that already contain Oracle VM Servers fail and result in an error.

Options

The following table shows the available options for this command.

Option	Description
`clusterEnable=` { `Yes` | `No` }	Whether to enable a clustered server pool. If this parameter is not included, the default is `No`, so the server pool is not clustered. If this parameter is set to `Yes`, you must also include either the `fileSystem` or `physicalDisk` option to provide a location for the server pool file system.
`clusterTimeout=` *value*	Set the timeout value for the cluster in seconds. *value* is an integer between 30 and 300.
`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }	The key mapping to be used when connecting to a virtual machine's console.
`migrateUsingSsl=` { `Yes` | `No` }	Whether to enable secure migration of virtual machines using SSL.
`masterServer=`*value*	The name or ID of the Oracle VM Server to use as the master server for the server pool.
`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` }	The policy by which virtual machines are located when created in the server pool. The default is `CURRENT_SERVER`.
`policyMode=` { `OFF` | `DRS` | `DPM` }	Set the policy to use for the server pool.
`policyCpuEnable=` { `Yes` | `No` }	Set whether to enable the policy set in the `policyMode` option for the server pool.
`policyPeriod=`*value*	The time period for the policy job to run. This sets the policy job to run every *n* minutes, for example, 10 sets the policy job to run every 10 minutes. *value* can be an integer between `2` and `60`.
`policyCpuThreshold=`*value*	The maximum amount of CPU percentage usage allowed before the policy must be enacted. *value* can be an integer between `0` and `99`.
`name=`*value*	A name to identify the server pool.
`description=`*value*	Optional description for the server pool. *value* is a maximum of 4,000 characters.

Option	Description
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.103 Editing a server pool

```
OVM> edit ServerPool name=MyServerPool name=MyOtherServerPool migrateUsingSsl=Yes
```

Example A.104 Changing the master Oracle VM Server

```
OVM> edit ServerPool id=0004fb000002000037db5e362c85a3fe masterServer=MyServer
```

Example A.105 Changing the virtual machine start policy

```
OVM> edit ServerPool name=MyServerPool startPolicy=BEST_SERVER
```

See Also

- Section A.43, “create ServerPool”
- Section A.44, “create ServerPoolNetworkPolicy”
- Section A.81, “edit ServerPoolNetworkPolicy”
- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.81 edit ServerPoolNetworkPolicy

Edits a server pool network policy.

Syntax

`edit ServerPoolNetworkPolicy` *`instance`* [`policyEnable=` { `Yes` | `No` }] [`policyThreshold=`*`value`*] `name=`*`value`* [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a server pool network policy.

Options

The following table shows the available options for this command.

Option	Description
`policyEnable=` { `Yes` | `No` }	Set whether to enable the policy set in the `policyMode` option of the create ServerPool or edit ServerPool command.
`policyThreshold=`*`value`*	The percentage (%) of network bandwidth the policy uses to move virtual machines. *`value`* can be an integer between 0 and 100.
`name=`*`value`*	A name to identify the server pool network policy.
`description=`*`value`*	Optional description for the server pool network policy. *`value`* is a maximum of 4,000 characters.

Examples

Example A.106 Editing a server pool network policy

```
OVM> edit ServerPoolNetworkPolicy name=MyNetworkPolicy policyEnable=No name="My Network Policy"
```

See Also

- Section A.44, “create ServerPoolNetworkPolicy”
- Section A.5, “addPolicyServer”
- Section A.130, “removePolicyServer”
- Section A.43, “create ServerPool”
- Section A.80, “edit ServerPool”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.82 edit ServerUpdateGroup

Edits an Oracle VM Server update group in a server pool.

Syntax

`edit ServerUpdateGroup` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits an Oracle VM Server update group in a server pool.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the server update group.
`description=`*`value`*	Optional description for the server update group. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.107 Editing the name of an update group for a server pool

```
OVM> edit ServerUpdateGroup name=MyServerUpdateGroup name="My Server Update Group"
```

See Also

- Section A.45, “create ServerUpdateGroup”
- Section A.46, “create ServerUpdateRepository”
- Section A.83, “edit ServerUpdateRepository”
- Section A.19, “checkUpToDate”
- Section A.160, “upgrade”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.83 edit ServerUpdateRepository

Edits an Oracle VM Server update repository.

Syntax

`edit ServerUpdateRepository` *`instance`* [`repositoryName=`*`value`*] [`url=`*`value`*] [`enabled=` { `Yes` | `No` }] [`pkgSignatureType=` { `NONE` | `GPG` | `CA` }] [`pkgSignatureKey=`*`value`*] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits an Oracle VM Server update repository.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`repositoryName=`*`value`*	A name for the server update repository. This may only contain alphanumeric characters and underscores. No spaces are permitted.
`url=`*`value`*	The URL to access the repository. If you enclose the URL in quotes, you must escape each forward slash (`/`) with another, for example: `url="http:////10.172.77.200//ovs"`
`enabled=` { `Yes` | `No` }	Whether to enable the repository.
`pkgSignatureType=` { `NONE` | `GPG` | `CA` }	The signature type to verify the validity of the repository, either `GPG` key (or GnuPG key), `CA` (Certificate Authority). Use `NONE` if there is no verification required.
`pkgSignatureKey=`*`value`*	The verification signature for the repository, for example, the location of the GPG key using any of the HTTP, FTP, FILE or HTTPS protocols. If you enclose the value for the option in quotes, you must escape each forward slash (`/`) with another, for example: `pkgSignatureKey="http:////10.172.77.200//ovs//RPM-GPG-KEY"`
`name=`*`value`*	A name to identify the server update repository.
`description=`*`value`*	Optional description for the server update repository. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.108 Editing an Oracle VM Server update repository

```
OVM> edit ServerUpdateRepository url=http://10.172.77.200/ovs enabled=No pkgSignatureType=GPG \
   pkgSignatureKey=http://10.172.77.200/ovs/RPM-GPG-KEY-oracle
```

See Also

- Section A.46, “create ServerUpdateRepository”
- Section A.45, “create ServerUpdateGroup”
- Section A.82, “edit ServerUpdateGroup”
- Section A.19, “checkUpToDate”
- Section A.160, “upgrade”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.84 edit StorageArray

Edits a storage array.

Syntax

`edit StorageArray` *`instance`* [`plugin=`*`value`*] [`storageName=`*`value`*] [`useChap=` { `Yes` | `No` }] [`adminHost=`*`value`* `adminUserName=`*`value`* `adminPassword=`*`value`*] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a storage array.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`plugin=`*`value`*	The name or ID for the Oracle VM Storage Connect plug-in for the storage array. To see a list of the available plug-ins, use the `list StorageArrayPlugin` command.
`storageName=`*`value`*	A name to identify the storage.
`useChap=` { `Yes` | `No` }	Whether to use CHAP authentication.
`adminHost=`*`value`*	The host name or IP address where administrative access to the storage array is allowed.
`adminUserName=`*`value`*	A user name with administrative access to the storage array, used with `adminHost`.
`adminPassword=`*`value`*	The administrator password for the `adminUserName` user.
`name=`*`value`*	A name to identify the storage array.
`description=`*`value`*	Optional description for the storage array. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyStorageArray`.

Examples

Example A.109 Editing the name of a storage array

```
OVM> edit StorageArray name=MyISCSIServer name="My ISCI Server"
```

See Also

- Section A.47, “create StorageArray”

- Section A.11, “add Server”
- Section A.136, “remove Server”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.161, “validate”
- Section A.124, “refresh”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.85 edit StorageArrayPlugin

Edits a storage array plug-in.

Syntax

`edit StorageArrayPlugin` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a storage array plug-in. You cannot create or delete a storage array plug-in; you can only edit the name and description.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the storage array plug-in.
`description=`*`value`*	Optional description for the storage array plug-in. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyStorageArrayPlugin`.

Examples

Example A.110 Editing a storage array plug-in

```
OVM> edit StorageArrayPlugin name="oracle.generic.SCSIPlugin.GenericPlugin (1.1.0)" \
  description="Generic storage array plug-in"
```

See Also

- Section A.161, “validate”
- Section A.121, “list”
- Section A.151, “show”

A.86 edit Tag

Edits a tag.

Syntax

`edit Tag` *`instance`* `name=`*`value`* [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a tag used to identify and group objects.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the tag.
`description=`*`value`*	Optional description for the tag. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyTag`.

Examples

Example A.111 Editing a tag

```
OVM> edit Tag name=MyTag name=MyNewTagName description="My new tag name."
```

See Also

- Section A.48, “create Tag”
- Section A.14, “add Tag”
- Section A.139, “remove Tag”
- Section A.57, “delete”
- Section A.121, “list”

- Section A.151, “show”

A.87 edit VirtualCdrom

Edits an ISO file/CDROM.

Syntax

`edit VirtualCdrom` *`instance`* [`sharable=` { `Yes` | `No` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an ISO file/CDROM.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`sharable=` { `Yes` | `No` }	Whether the ISO file/CDROM may be shared.
`name=`*`value`*	A name to identify the ISO file/CDROM.
`description=`*`value`*	Optional description for the ISO file/CDROM. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyISOFile`.

Examples

Example A.112 Editing a ISO file/CDROM

```
OVM> edit VirtualCdrom id=0004fb0000150000cd7223d8105042c5.iso name="OL6" \
  description="Oracle Linux 6"
```

See Also

- Section A.117, “importVirtualCdrom”
- Section A.25, “cloneCdToRepo”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.88 edit VirtualDisk

Edits a virtual disk.

Syntax

`edit VirtualDisk` *`instance`* [`shareable=` { `Yes` | `No` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a virtual disk in a storage repository. To resize a virtual disk, use the resize command.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`shareable=` { `Yes` | `No` }	Whether the virtual disk is shareable. Shareable disks have read/write privileges in multiple virtual machines and should be used with caution.
`name=`*`value`*	A name to identify the virtual disk.
`description=`*`value`*	Optional description for the virtual disk. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVirtualDisk`.

Examples

Example A.113 Editing a virtual disk

```
OVM> edit VirtualDisk name=MyVMDisk name='New name for MyVMDisk' description='My virtual disk'
```

See Also

- Section A.118, “importVirtualDisk”
- Section A.49, “create VirtualDisk”
- Section A.143, “resize”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.89 edit VlanInterface

Edits a VLAN interface.

Syntax

`edit VlanInterface` *`instance`* [`mtu=`*`value`*] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a VLAN interface.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`mtu=`*`value`*	The MTU value. May be an integer between `1500` and `64000`.
`name=`*`value`*	A name to identify the VLAN interface.
`description=`*`value`*	Optional description for the VLAN interface. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVlanInterface`.

Examples

Example A.114 Editing a VLAN interface

```
OVM> edit VlanInterface name=MyVlanInterface mtu=1500
```

See Also

- Section A.50, “create VlanInterface”
- Section A.15, “add VlanInterface”
- Section A.140, “remove VlanInterface”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.90 edit Vm

Edits a virtual machine.

Syntax

`edit Vm` *`instance`* [`memory=`*`value`*] [`memoryLimit=`*`value`*] [`cpuCount=`*`value`*] [`cpuCountLimit=`*`value`*] [`cpuPriority=`*`value`*] [`cpuUtilizationCap=`*`value`*] [`highAvailability=` { `Yes` | `No` }] [`hugePagesEnabled=` { `Yes` | `No` }] [`osType=`*`value`*] [`mouseType=` { `OS_DEFAULT` | `PS2_MOUSE` | `USB_MOUSE` | `USB_TABLET` }] [`domainType=` { `XEN_HVM` | `XEN_HVM_PV_DRIVERS` | `XEN_PVM` | `LDOMS_PVM` | `UNKNOWN` }] [`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }] [`bootOrder=` { `PXE` | `DISK` | `CDROM` }] [`networkInstallpath=`*`value`*] [`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` | `USE_POOL_POLICY` }] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a virtual machine to change the configuration options.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`memory=`*`value`*	The memory size the virtual machine is allocated in MB. May be an integer between `32` and `2000000`.
`memoryLimit=`*`value`*	The maximum memory size the virtual machine can be allocated in MB. May be an integer between `32` and `2000000`.
`cpuCount=`*`value`*	The number of processors the virtual machine is allocated. May be an integer between `1` and `128`.
`cpuCountLimit=`*`value`*	The maximum number of processors the virtual machine can be allocated. May be an integer between `1` and `128`.
`cpuPriority=`*`value`*	The CPU priority of the virtual machine. A value between `1` and `100`; the higher the number, the more priority the CPU is given.
`cpuUtilizationCap=`*`value`*	The maximum percentage to which the virtual CPUs can receive scheduled time. A value between `10` and `100`; the higher the number, the more scheduled time the CPU is given.
`highAvailability=` { `Yes` | `No` }	Whether to enable High Availability (HA).
`hugePagesEnabled=` { `Yes` | `No` }	Whether to enable HugePages. If you enable HugePages, you should also enable support for HugePages in the guest operating system. This option must only be used for virtual machines that have their domain type set to XEN_PVM. More information on Huge Page support is provided in How are Huge Pages Enabled for Virtual Machines? in the *Oracle VM Concepts Guide*.

Option	Description
`osType=` *value*	The operating system of the virtual machine. To find the operating system type, use the getVmOsTypes command.
`mouseType=` { `OS_DEFAULT` | `PS2_MOUSE` | `USB_MOUSE` | `USB_TABLET` }	The mouse type of the virtual machine.
`domainType=` { `XEN_HVM` | `XEN_HVM_PV_DRIVERS` | `XEN_PVM` | `LDOMS_PVM` | `UNKNOWN` }	The domain type of the virtual machine.
`keymapName=` { `en-us` | `ar` | `da` | `de` | `de-ch` | `en-gb` | `es` | `et` | `fi` | `fo` | `fr` | `fr-be` | `fr-ca` | `fr-ch` | `hr` | `hu` | `is` | `it` | `ja` | `lt` | `lv` | `mk` | `nl` | `nl-be` | `No` | `pl` | `pt` | `pt-br` | `ru` | `sl` | `sv` | `th` | `tr` }	The keyboard mapping to use for the virtual machine.
`bootOrder=` { `PXE` | `DISK` | `CDROM` }	The boot media order for the virtual machine. Enter options separated by commas (,), for example: `bootOrder='CDROM,DISK'` If you use the `PXE` boot option to boot from network-based installation media, also use the `networkInstallpath` parameter.
`networkInstallpath=`*value*	The location at which the installation media (mounted ISO file) is located when creating a PVM guest.
`startPolicy=` { `BEST_SERVER` | `CURRENT_SERVER` | `USE_POOL_POLICY` }	Optional virtual machine start up policy.
`name=`*value*	A name to identify the virtual machine.
`description=`*value*	Optional description for the virtual machine. *value* is a maximum of 4,000 characters.
{ `id=`*value* | `name=`*value* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.115 Edit a virtual machine

```
OVM> edit Vm name=MyVM bootOrder='CDROM,DISK' startPolicy=BEST_SERVER
```

See Also

- Section A.110, “getVmOsTypes”
- Section A.24, “clone Vm”
- Section A.16, “add Vm”
- Section A.156, “start”
- Section A.121, “list”
- Section A.151, “show”

A.91 edit VmCloneCustomizer

Edits an existing clone customizer for a virtual machine.

Syntax

`edit VmCloneCustomizer` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command allows you to edit an existing clone customizer for a virtual machine.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name to identify the clone customizer. **Tip** To find this name after the clone customizer has been created on a virtual machine, use the `list VmCloneCustomizer` command. You need the name or ID of this to delete a clone customizer from a virtual machine with the `delete VmCloneCustomizer` command.
`description=`*`value`*	Optional description for the clone customizer object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVMCloneCustomizer`.

Examples

Example A.116 Edit a virtual machine clone customizer

```
OVM> edit VmCloneCustomizer name=MyVmCloneCustomizer name=MyCloneCustomizer \
description="A test clone customizer"
```

See Also

- Section A.93, “edit VmCloneStorageMapping”
- Section A.92, “edit VmCloneNetworkMapping”
- Section A.52, “create VmCloneCustomizer”

- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.92 edit VmCloneNetworkMapping

Edits an existing network mapping for a virtual machine clone customizer.

Syntax

`edit VmCloneNetworkMapping` *`instance`* [`network=`*`value`*] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

Edits an existing network mapping for a virtual machine clone customizer.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`network=`*`value`*	The name or ID value of an existing network.
`name=`*`value`*	A name to identify the clone network mapping.
`description=`*`value`*	Optional description for the clone network mapping object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyCloneNet`.

Examples

Example A.117 Editing a virtual machine clone network mapping

```
OVM> edit VmCloneNetworkMapping name=MyCloneNet description="A clone network mapping"
```

See Also

- Section A.52, “create VmCloneCustomizer”
- Section A.53, “create VmCloneNetworkMapping”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.93 edit VmCloneStorageMapping

Edits an existing clone customizer storage mapping.

Syntax

`edit VmCloneStorageMapping` *`instance`* [`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }] { `physicalDisk=`*`value`* | `repository=`*`value`* | `storageArray=`*`value`* } [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

Edits an existing clone customizer storage mapping.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`cloneType=` { `SPARSE_COPY` | `NON_SPARSE_COPY` | `THIN_CLONE` }	The type of clone storage that should be created.
{ `physicalDisk=`*`value`* | `repository=`*`value`* | `storageArray=`*`value`* }	The name or ID of either a physicalDisk, repository or storageArray object where the clone disk is located.
`name=`*`value`*	A name to identify the clone storage mapping.
`description=`*`value`*	Optional description for the clone storage mapping object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVMCloneStorage`.

Examples

Example A.118 Editing a virtual machine clone storage mapping

```
OVM> edit VmCloneStorageMapping name=MyVMCloneStorage cloneType=THIN_CLONE
```

See Also

- Section A.52, “create VmCloneCustomizer”
- Section A.93, “edit VmCloneStorageMapping”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.94 edit VmDiskMapping

Edits the virtual machine disk mapping object.

Syntax

`edit VmDiskMapping` *`instance`* [`virtualCd=`*`instance`*] [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the virtual machine disk mapping object. Do not use this command to change the disk mapping in a virtual machine, this is just for the object that contains the disk mapping information. To edit a virtual disk or CDROM mapped to a virtual machine, remove it using the `delete VmDiskMapping` command, then use the create VmDiskMapping command again to remap it to a virtual machine with any changed settings.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`virtualCd=`*`instance`*	The name or ID of the ISO file (virtual CDROM).
`name=`*`value`*	A name to identify the disk mapping. **Tip** To find this name after a virtual disk is mapped to a virtual machine, use the `list VmDiskMapping` command.
`description=`*`value`*	Optional description for the disk mapping object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDiskMap`.

Examples

Example A.119 Editing a virtual disk mapping object

```
OVM> edit VmDiskMapping id=0004fb0000130000409cd9340443e257 name=MyDiskMap
```

See Also

- Section A.55, “create VmDiskMapping”
- Section A.57, “delete”
- Section A.121, “list”

- Section A.151, “show”

A.95 edit Vnic

Edits a VNIC.

Syntax

`edit Vnic` *`instance`* [`name=`*`value`*] [`description=`*`value`*] [`macAddress=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a VNIC on a network.

To change the network assignment of a VNIC, please see Section A.17, “add Vnic”.

Options

The following table shows the available options for this command.

Option	Description
`name=`*`value`*	A name for the VNIC.
`description=`*`value`*	Optional description for the VNIC. *`value`* is a maximum of 4,000 characters.
`macAddress=`*`value`*	The MAC address that should be applied to this VNIC.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVNIC`.

Examples

Example A.120 Editing a VNIC

```
OVM> edit Vnic id=0004fb00000700007fa68ffd2011539f name=Server1Vnic macAddress=00:21:f6:00:00:18
```

See Also

- Section A.56, “create Vnic”
- Section A.17, “add Vnic”
- Section A.142, “remove Vnic”
- Section A.150, “setVnicMacAddrRange”
- Section A.113, “getVnicMacAddrRange”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.96 edit VolumeGroup

Edits a volume group object.

Syntax

`edit VolumeGroup` *`instance`* [`name=`*`value`*] [`description=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of a volume group object.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	**Description**
`network=`*`value`*	The name or ID of the volume group object.
`name=`*`value`*	A name for the volume group object.
`description=`*`value`*	Optional description for the volume group object. *`value`* is a maximum of 4,000 characters.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVolumeGroup`.

Examples

Example A.121 Editing a volume group

```
OVM> edit VolumeGroup name=MyVolumeGroup name=MyNewName
```

See Also

- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.97 embeddedCreate

Creates an IP address object and adds it to a network related object.

Syntax

`embeddedCreate` { `BondPort` | `Port` | `VlanInterface` } *`instance`* `ipAddressConfig` `ipAddressConfigType=` { `STATIC` | `DYNAMIC` } [`ipAddress=`*`value`*] [`ipNetmask=`*`value`*]

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command adds an IP address and its associated configuration (ipAddressConfig object) to a bond port, port, or VLAN interface.

Options

The following table shows the available options for this command.

Option	Description
{ BondPort | Port | VlanInterface }	The network related object to which to add the IP address (ipAddressConfig object).
ipAddressConfigType= { STATIC | DYNAMIC }	The IP addressing configuration type.
[ipAddress=*value*]	The IP address.
[ipNetmask=*value*]	The netmask.
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyPort.

Examples

Example A.122 Adding an IP address to a port

```
OVM> embeddedCreate Port id=0004fb00002000003ea1bffb91ece960 ipAddressConfig \
  ipAddressConfigType=STATIC ipAddress=10.172.76.100 ipNetmask=255.255.254.0
```

Example A.123 Adding an IP address to a bond port

```
OVM> embeddedCreate BondPort id=0004fb00002000000a5389824228bdf1 ipAddressConfig \
  ipAddressConfigType=STATIC ipAddress=10.172.76.100 ipNetmask=255.255.254.0
```

See Also

- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.121, “list”
- Section A.151, “show”

A.98 embeddedDelete

Deletes an IP address object and removes it from a network related object.

Syntax

embeddedDelete { BondPort | Port | VlanInterface } *instance* ipAddressConfig id=*value*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command removes an IP address and its associated configuration (ipAddressConfig object) from a bond port, port, or VLAN interface.

Options

The following table shows the available options for this command.

Option	Description
{ BondPort | Port | VlanInterface }	The network related object from which to remove the IP address (ipAddressConfig object).

id=*value*	The index value of the ipAddressConfig object. To find the index value, use the show command and look for the embedded object's IP address information. For example, to find the index value for an IP address object on a port:
	`OVM> show Port id=0004fb00002000003ea1bffb91ece960`
	`...`
	`Ip Address Config 1 - Address Type = Ipv4`
	`Ip Address Config 1 - Config Type =  Static`
	`Ip Address Config 1 - Address = 10.172.76.100`
	`Ip Address Config 1 - Netmask = 255.255.254.0`
	`Interface Name = eth2`
	`...`
	The index value in this case is 1.

{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyPort.

Examples

Example A.124 Deleting an IP address from a port

```
OVM> embeddedDelete Port id=0004fb00002000003ea1bffb91ece960 ipAddressConfig id=1
```

Example A.125 Deleting an IP address from a bond port

```
OVM> embeddedDelete BondPort id=0004fb00002000000a5389824228bdf1 ipAddressConfig id=1
```

See Also

- Section A.97, “embeddedCreate”
- Section A.99, “embeddedEdit”
- Section A.121, “list”
- Section A.151, “show”

A.99 embeddedEdit

Edits an IP address object on a network related object.

Syntax

`embeddedEdit` { `BondPort` | `Port` | `VlanInterface` } *`instance`* `ipAddressConfig` `id=`*`value`* `ipAddressConfigType=` { `STATIC` | `DYNAMIC` } [`ipAddress=`*`value`*] [`ipNetmask=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command edits the attributes of an IP address and its associated configuration (ipAddressConfig object) on a bond port, port, or VLAN interface.

Options

The following table shows the available options for this command.

Option	Description
{ `BondPort` | `Port` | `VlanInterface` }	The network related object on which to edit the IP address (ipAddressConfig object).

`id=`*`value`*	The index value of the ipAddressConfig object. To find the index value, use the show command and look for the embedded object's IP address information. For example, to find the index value for an IP address object on a port:
	`OVM> show Port id=0004fb00002000003ea1bffb91ece960`
	`...`
	`Ip Address Config 1 - Address Type = Ipv4`
	`Ip Address Config 1 - Config Type =  Static`
	`Ip Address Config 1 - Address = 10.172.76.100`
	`Ip Address Config 1 - Netmask = 255.255.254.0`
	`Interface Name = eth2`
	`...`
	The index value in this case is `1`.

`ipAddressConfigType=` { `STATIC` | `DYNAMIC` }	The IP addressing configuration type.
[`ipAddress=`*`value`*]	The IP address.
[`ipNetmask=`*`value`*]	The netmask.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyPort`.

Examples

Example A.126 Editing an IP address on a port

```
OVM> embeddedEdit Port id=0004fb00002000003ea1bffb91ece960 ipAddressConfig \
  id=1 ipAddressConfigType=STATIC ipAddress=10.172.76.100 ipNetmask=255.255.254.0
```

Example A.127 Editing an IP address on a bond port

```
OVM> embeddedEdit BondPort id=0004fb00002000000a5389824228bdf1 ipAddressConfig \
  id=1 ipAddressConfigType=STATIC ipAddress=10.172.76.100 ipNetmask=255.255.254.0
```

See Also

- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.121, “list”
- Section A.151, “show”

A.100 exit

Exits/Quits the CLI.

Syntax

`exit`

Description

This command exits/quits the CLI.

Options

This command does not take any arguments or provide any options.

Examples

Example A.128 Exiting the CLI

```
OVM> exit
```

See Also

- Section A.114, “help”

A.101 getDebugTranscript

Shows the debug transcript of a job.

Syntax

`getDebugTranscript` `Job` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command shows shows the debug transcript of a job.

Options

The following table shows the available options for this command.

Option	Description
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyJob.

Examples

Example A.129 Showing the debug transcript of a job

```
OVM> getDebugTranscript Job id=1373345941846
```

See Also

- Section A.1, “abort Job”
- Section A.105, “getJobs”
- Section A.121, “list”
- Section A.151, “show”

A.102 getDescriptor

Lists the descriptor file for an assembly.

Syntax

getDescriptor Assembly *instance*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command lists the descriptor file for an assembly. The descriptor (.ovf file) is the main file in an assembly package and contains meta-data for the assembly, including links to external files, such as virtual disks.

Options

The following table shows the available options for this command.

Option	Description
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyAssembly.

Examples

Example A.130 Listing the descriptor for an assembly

```
OVM> getDescriptor Assembly name=MyAssembly.ova
```

See Also

- Section A.115, “importAssembly”

- Section A.31, “createVmFromAssembly”
- Section A.24, “clone Vm”
- Section A.61, “edit Assembly”
- Section A.62, “edit AssemblyVirtualDisk”
- Section A.63, “edit AssemblyVm”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.103 getEvents

Lists the *events* for an object.

Syntax

`getEvents` [[`objType=`*`value`*] [`objId=`*`value`*]] [`severity=` { `UNKNOWN` | `CRITICAL` | `WARNING` | `INFORMATIONAL` }] [`acknowledged=` { `Yes` | `No` }] [`startTime=`*`value`*] [`endTime=`*`value`*] [`amount=`*`value`*]

Description

This command lists the event for an object.

Options

The following table shows the available options for this command. To acknowledge an event, use the ackEvent command.

Option	Description
`objType=`*`value`*	The object type for which to list the events. The *`value`* of `objType` is an object such as `FileServer`, `Repository`, or `Server`. To get a list of all object types, use the showobjtypes. The value for `objType` is case insensitive. This is an optional parameter and must be used with the `objId` option.
`objId=`*`value`*	The name or ID of the object. This is an optional parameter and must be used with the `objType` option.
`severity=` { `UNKNOWN` | `CRITICAL` | `WARNING` | `INFORMATIONAL` }	The severity of the event to list. This is an optional parameter.
`acknowledged=` { `Yes` | `No` }	Whether to include acknowledged events in the listing. The default is *`No`*, so all events that have been acknowledged are not displayed.

Option	Description
	This is an optional parameter.
`startTime=value`	The start date and time from which to list events. The format to use is `"MM-dd-yyyy HH:mm"`. This is an optional parameter.
`endTime=value`	The end date and time from which to list events. The format to use is `"MM-dd-yyyy HH:mm"`. This is an optional parameter.
`amount=value`	The maximum number of events to list. This is an optional parameter.

Examples

Example A.131 Listing all events

```
OVM> getEvents
```

Example A.132 Listing all critical error events

```
OVM> getEvents severity=CRITICAL
```

Example A.133 Listing Oracle VM Server events using all options

```
OVM> getEvents objType=Server objId=00:e0:81:4d:40:c6:00:e0:81:4d:40:c7:ff:ff:ff:ff \
  severity=UNKNOWN acknowledged=Yes startTime="05-20-2013 00:00" \
  endTime="05-21-2013 23:59" amount=100
```

Example A.134 Listing all unacknowledged critical error events for a virtual machine

```
OVM> getEvents objType=Vm objId=0004fb00001400003f45fc117b56c135 severity=CRITICAL \
  acknowledged=No
```

See Also

- Section A.154, “showobjtypes”
- Section A.104, “getEventsForObject”
- Section A.2, “ackEvent”

A.104 getEventsForObject

Lists the *events* for an object.

Syntax

`getEventsForObject objType=value objId=value`

Description

This command lists the events for an object. To acknowledge an event, use the ackEvent command.

Options

The following table shows the available options for this command.

Option	Description
`objType=value`	The object type for which to list the events. The *value* of `objType` is an object such as `FileServer`, `Repository`, or `Server`. To get a list of all object types, use the showobjtypes command. The value for `objType` is case insensitive.
`objId=value`	The name or ID of the object.

Examples

Example A.135 Listing events for an Oracle VM Server

```
OVM> getEventsForObject objType=Server objId=MyServer
```

See Also

- Section A.154, "showobjtypes"
- Section A.103, "getEvents"
- Section A.108, "getTriageEvent"
- Section A.2, "ackEvent"

A.105 getJobs

Lists *jobs*.

Syntax

`getJobs` [`startTime=value`] [`endTime=value`] [`amount=value`]

Description

This command lists all *jobs*, or jobs within a date range.

Although none of the options are mandatory, you must supply at least one option.

Options

The following table shows the available options for this command.

Option	Description
`startTime=value`	The start date and time from which to list jobs. The format to use is `"MM-dd-yyyy HH:mm"`.
`endTime=value`	The end date and time from which to list jobs. The format to use is `"MM-dd-yyyy HH:mm"`.
`amount=value`	The number of jobs to list.

Examples

Example A.136 Listing jobs in a date range

```
OVM> getJobs startTime="07-20-2012 12:00" endTime="07-22-2012 24:00"
```

See Also

- Section A.106, “getQueuedJobInfo”
- Section A.101, “getDebugTranscript”
- Section A.1, “abort Job”
- Section A.121, “list”
- Section A.151, “show”

A.106 getQueuedJobInfo

Lists information about a queued job.

Syntax

`getQueuedJobInfo` `Job` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command lists information about a queued job.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyJob`.

Examples

Example A.137 Listing information about a queued job

```
OVM> getQueuedJobInfo Job id=1394647459371
```

See Also

- Section A.105, “getJobs”
- Section A.101, “getDebugTranscript”
- Section A.1, “abort Job”

- Section A.121, “list”
- Section A.151, “show”

A.107 getStatsConfig

Shows the configuration for the statistics displayed in Oracle VM Manager.

Syntax

`getStatsConfig`

Description

This command shows the configuration for the statistics displayed in Oracle VM Manager.

Options

This command does not take any arguments or provide any options.

Examples

Example A.138 Showing the Oracle VM Manager statistics configuration

```
OVM> getStatsConfig
```

See Also

- Section A.149, “setStatsConfig”

A.108 getTriageEvent

Lists the highest severity *event* for an object.

Syntax

`getTriageEvent objType=`*`value`* `objId=`*`value`*

Description

This command lists the highest severity event for an object. To acknowledge an event, use the ackEvent command.

Options

The following table shows the available options for this command.

Option	Description
`objType=`*`value`*	The object type for which to list the event. The *`value`* of `objType` is an object such as `FileServer`, `Repository`, or `Server`. To get a list of all object types, use the showobjtypes command. The value for `objType` is case insensitive.
`objId=`*`value`*	The name or ID of the object.

Examples

Example A.139 Listing the highest severity event for an Oracle VM Server

```
OVM> getTriageEvent objType=Server objId=MyServer
```

See Also

- Section A.154, “showobjtypes”
- Section A.103, “getEvents”
- Section A.104, “getEventsForObject”
- Section A.2, “ackEvent”

A.109 getVmCfgFileContent

Shows the contents of a virtual machine configuration file.

Syntax

`getVmCfgFileContent` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command shows the contents of a virtual machine configuration file (`vm.cfg`).

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVm`.

Examples

Example A.140 Showing the contents of a virtual machine configuration file

```
OVM> getVmCfgFileContent Vm name=MyVM
```

See Also

- Section A.121, “list”
- Section A.151, “show”

A.110 getVmOsTypes

Shows the virtual machine operating system types.

Syntax

`getVmOsTypes`

Description

This command shows the virtual machine operating types used when creating or editing a virtual machine.

Options

This command does not take any arguments or provide any options.

Examples

Example A.141 Showing virtual machine operating system types

```
OVM> getVmOsTypes
```

See Also

- Section A.51, “create Vm”
- Section A.90, “edit Vm”

A.111 getVmReceivedMessages

Lists the key/value pair messages received by a running virtual machine.

Syntax

`getVmReceivedMessages` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command lists the key/value pair messages received by a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.142 Listing messages received by a virtual machine

```
OVM> getVmReceivedMessages Vm name=MyVm
```

See Also

- Section A.146, “sendVmMessage”
- Section A.112, “getVmSentMessages”
- Section A.22, “clearVmRcvdMessage”
- Section A.23, “clearVmSentMessage”
- Section A.21, “clearVmAllSentMessages”
- Section A.20, “clearVmAllRcvdMessages”

A.112 getVmSentMessages

Lists the key/value pair messages sent to a running virtual machine.

Syntax

`getVmSentMessages` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command lists the key/value pair messages sent to a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.143 Listing messages sent to a virtual machine

```
OVM> getVmSentMessages Vm name=MyVm
```

See Also

- Section A.146, “sendVmMessage”
- Section A.111, “getVmReceivedMessages”
- Section A.22, “clearVmRcvdMessage”
- Section A.23, “clearVmSentMessage”
- Section A.21, “clearVmAllSentMessages”

- Section A.20, “clearVmAllRcvdMessages”

A.113 getVnicMacAddrRange

Displays the range of MAC addresses that are available to VNICs.

Syntax

```
getVnicMacAddrRange
```

Description

This command displays the range of MAC addresses that can be used when creating a VNIC. To set the MAC address range, use the setVnicMacAddrRange command.

Options

This command does not take any arguments or provide any options.

Examples

Example A.144 Displaying the VNIC MAC Address Range

```
OVM> getVnicMacAddrRange
```

See Also

- Section A.150, “setVnicMacAddrRange”
- Section A.56, “create Vnic”
- Section A.95, “edit Vnic”
- Section A.17, “add Vnic”
- Section A.142, “remove Vnic”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.114 help

Provides a list of commonly used commands and their syntax.

Syntax

```
help
```

Description

This command provides a list of commonly used commands and their syntax. It groups the commands according to how they are generally used.

Options

This command does not take any arguments or provide any options.

Examples

Example A.145 Using the Help Command

```
OVM> help
```

See Also

- Section A.152, “showallcustomcmds”
- Section A.153, “showcustomcmds”
- Section A.154, “showobjtypes”

A.115 importAssembly

Imports and adds an *assembly* to a storage repository.

Syntax

`importAssembly` `Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command imports and adds an assembly file to a storage repository. The imported assembly is unpacked and each virtual machine is contained within an AssemblyVm object. The AssemblyVm objects are created in the same storage repository as the original assembly file. Use the `list AssemblyVm` command to find the name and ID of the new AssemblyVm objects, then use the createVmFromAssembly command to create a virtual machine templates from each AssemblyVm object.

Options

The following table shows the available options for this command.

Option	Description
`Repository` *`instance`*	The storage repository in which to import the assembly.
`url=`*`value`*	The URL of the assembly. Note that if you quote this argument, the forward slashes in the URL should be escaped. This is illustrated in the example.
`proxy=`*`value`*	The IP address or hostname of a proxy server to use when importing the assembly.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.146 Importing an assembly to a storage repository

```
OVM> importAssembly Repository name=MyRepository url="http:////example.com//myassembly.ova"
```

See Also

- Section A.31, “createVmFromAssembly”
- Section A.61, “edit Assembly”
- Section A.62, “edit AssemblyVirtualDisk”
- Section A.63, “edit AssemblyVm”
- Section A.102, “getDescriptor”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”

A.116 importTemplate

Imports and adds a virtual machine template to a storage repository.

Syntax

`importTemplate` `Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command imports and adds a virtual machine template to a storage repository.

Virtual machines and virtual machine templates are treated the same in the CLI, so many of the commands you use to manage templates are handled by the same commands as managing virtual machines, for example, to list templates, use the `list Vm` command, and to delete a template, use the `delete Vm` command.

Options

The following table shows the available options for this command.

Option	Description
`Repository` *`instance`*	The storage repository in which to import the virtual machine template.

Option	Description
`url=`*`value`*	The URL of the virtual machine template. Note that if you quote this argument, the forward slashes in the URL should be escaped. This is illustrated in the example.
`proxy=`*`value`*	The IP address or hostname of a proxy server to use when importing the template.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyRepository`.

Examples

Example A.147 Importing a virtual machine template to a storage repository

```
OVM> importTemplate Repository name=MyRepository url="http:////example.com//mytemplate.tgz"
```

See Also

- Section A.119, “importVirtualMachine”
- Section A.24, “clone Vm”
- Section A.124, “refresh”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.117 importVirtualCdrom

Imports and adds a virtual CDROM/ISO file to a storage repository.

Syntax

`importVirtualCdrom Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command imports and adds a virtual CDROM/ISO file to a storage repository.

Options

The following table shows the available options for this command.

Option	Description
`Repository` *`instance`*	The storage repository in which to import the virtual CDROM/ ISO file.

Option	Description
`url=`*`value`*	The URL of the virtual CDROM/ISO file. Note that if you quote this argument, the forward slashes in the URL should be escaped. This is illustrated in the example.
`proxy=`*`value`*	The IP address or hostname of a proxy server to use when importing the virtual CDROM/ISO file.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyRepository`.

Examples

Example A.148 Importing a virtual CDROM/ISO file to a storage repository

```
OVM> importVirtualCdrom Repository name=MyRepository url="http:////example.com//myiso.iso"
```

Tip

Note that, in the above example, forward slashes are used to escape the forward slashes that appear in the URL. Hence the doubling up of forward slashes.

See Also

- Section A.87, “edit VirtualCdrom”
- Section A.25, “cloneCdToRepo”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.118 importVirtualDisk

Imports and adds a virtual disk file to a storage repository.

Syntax

`importVirtualDisk Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command imports and adds a virtual disk file to a storage repository.

Options

The following table shows the available options for this command.

Option	Description
`Repository` *`instance`*	The storage repository in which to import the virtual disk file.

Option	Description
`url=`*`value`*	The URL of the virtual disk file. Note that if you quote this argument, the forward slashes in the URL should be escaped. This is illustrated in the example.
`proxy=`*`value`*	The IP address or hostname of a proxy server to use when importing the virtual disk file.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyRepository`.

Examples

Example A.149 Importing a virtual disk file to a storage repository

```
OVM> importVirtualDisk Repository name=MyRepository url="http:////example.com//myvdisk.img"
```

See Also

- Section A.49, “create VirtualDisk”
- Section A.88, “edit VirtualDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.119 importVirtualMachine

Imports and adds a virtual machine to a storage repository.

Syntax

`importVirtualMachine` `Repository` *`instance`* `url=`*`value`* [`proxy=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command imports and adds a virtual machine to a storage repository. The virtual machine is placed in the **Unassigned Virtual Machines** folder in Oracle VM Manager. To deploy the virtual machine to an Oracle VM Server, use the add Vm command.

Options

The following table shows the available options for this command.

Option	Description
`Repository` *`instance`*	The storage repository in which to import the virtual machine.

Option	Description
`url=value`	The URL of the virtual machine. To import a multi-file virtual machine, enter each URL in a comma separated list, for example: `url=http://example.com/Sys.img,http://example.com/vm.cfg` Note that if you quote this argument, the forward slashes in the URL should be escaped. This is illustrated in the example.
`proxy=value`	The IP address or hostname of a proxy server to use when importing the virtual machine.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyRepository`.

Examples

Example A.150 Importing a virtual machine to a storage repository

```
OVM> importVirtualMachine Repository name=MyRepository url="http:////example.com//mytemplate.tgz"
```

See Also

- Section A.16, “add Vm”
- Section A.116, “importTemplate”
- Section A.24, “clone Vm”
- Section A.124, “refresh”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.120 kill

Kills an Oracle VM Server or virtual machine.

Syntax

`kill` { `Server` | `Vm` } *`instance`*

Where *`instance`* is:

{ `id=value` | `name=value` }

Description

This command kills an Oracle VM Server or virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `Server` | `Vm` }	The object to kill, either an Oracle VM Server or a virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.151 Killing an Oracle VM Server

```
OVM> kill Server name=MyServer
```

Example A.152 Killing a virtual machine

```
OVM> kill Vm name=MyVM
```

See Also

- Section A.156, “start”
- Section A.144, “restart”
- Section A.157, “stop”

A.121 list

Lists all instances of an object.

Syntax

`list` { `AccessGroup` | `AntiAffinityGroup` | `Assembly` | `AssemblyVirtualDisk` | `AssemblyVm` | `BondPort` | `ControlDomain` | `Cpu` | `CpuCompatibilityGroup` | `FileServer` | `FileServerPlugin` | `FileSystem` | `Job` | `Manager` | `Network` | `PeriodicTask` | `PhysicalDisk` | `Port` | `Repository` | `RepositoryExport` | `Server` | `ServerController` | `ServerPool` | `ServerPoolNetworkPolicy` | `ServerUpdateGroup` | `ServerUpdateRepository` | `StorageArray` | `StorageArrayPlugin` | `StorageInitiator` | `Tag` | `VirtualCdrom` | `VirtualDisk` | `VlanInterface` | `Vm` | `VmCloneCustomizer` | `VmCloneNetworkMapping` | `VmCloneStorageMapping` | `VmDiskMapping` | `Vnic` | `VolumeGroup` }

Description

This command lists all instances of an object.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `AntiAffinityGroup` | `Assembly` | `AssemblyVirtualDisk` | `AssemblyVm` | `BondPort` | `ControlDomain` | `Cpu` | `CpuCompatibilityGroup` |	The object to list. The `Vm` option lists both virtual machines and virtual machine templates. You can use the `show` command with either the

Option	Description
FileServer | FileServerPlugin | FileSystem | Job | Manager | Network | PeriodicTask | PhysicalDisk | Port | Repository | RepositoryExport | Server | ServerController | ServerPool | ServerPoolNetworkPolicy | ServerUpdateGroup | ServerUpdateRepository | StorageArray | StorageArrayPlugin | StorageInitiator | Tag | VirtualCdrom | VirtualDisk | VlanInterface | Vm | VmCloneCustomizer | VmCloneNetworkMapping | VmCloneStorageMapping | VmDiskMapping | Vnic | VolumeGroup }	Repository or Vm option to distinguish between a virtual machine and a virtual machine template. The VmDiskMapping option lists the disk mapping objects for both virtual machines and virtual machine templates.

Examples

Example A.153 Listing Oracle VM Servers

```
OVM> list Server
```

Example A.154 Listing virtual machines and virtual machine templates

```
OVM> list Vm
```

Example A.155 Listing networks

```
OVM> list Network
```

Example A.156 Listing virtual machine and virtual machine templates disk mapping

```
OVM> list VmDiskMapping
```

See Also

- Section A.151, “show”

A.122 migrate Vm

Migrates a virtual machine.

Syntax

`migrate Vm` *`instance`* [`destServer=`*`value`* | `destServerPool=`*`value`*]

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command migrates a virtual machine to an Oracle VM Server or server pool.

You can *migrate* a running virtual machine to an Oracle VM Server within the same server pool. To migrate the virtual machine to the **Unassigned Virtual Machines** folder (undeploy it), do not supply a destination.

It is not possible to migrate a stopped virtual machine using this command. To do this, you should use the remove Vm and add Vm commands instead.

Options

The following table shows the available options for this command.

Option	**Description**
[`destServer=`*`value`* | `destServerPool=`*`value`*]	The name or ID of the Oracle VM Server or server pool on which to migrate the virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.157 Migrating a virtual machine to an Oracle VM Server

```
OVM> migrate Vm name=MyVM destServer=MyServer
```

Example A.158 Migrating a virtual machine to a server pool

```
OVM> migrate Vm name=MyVM destServerPool=MyServerPool
```

Example A.159 Migrating a virtual machine to the Unassigned Virtual Machines folder

```
OVM> migrate Vm name=MyVM
```

See Also

- Section A.16, “add Vm”
- Section A.141, “remove Vm”
- Section A.156, “start”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.123 moveVmToRepository

Moves a virtual machine to a storage repository that has been defined within a clone customizer.

Syntax

`moveVmToRepository Vm` *`instance`* `cloneCustomizer=`*`value`* `targetRepository=`*`value`*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command moves a virtual machine to a target storage repository. The target storage repository is set using a predefined clone customizer. Use the create VmCloneCustomizer command to create a clone customizer.

Any VmDiskMapping objects are renamed during the move operation. The move job copies any virtual disks to the target storage repository, and, consequently, the new virtual disks have new UUIDs. New VmDiskMapping objects are created to map the newly created virtual disks and added to the virtual machine. The old VmDiskMapping objects are then deleted. Deleting the VmDiskMapping object also causes any associated VmCloneStorageMapping objects to be deleted.

Options

The following table shows the available options for this command.

Option	Description
cloneCustomizer=*value*	The name or ID of the clone customizer to use to move the virtual machine to a storage repository.
targetRepository=*value*	The name or ID of the storage repository to which the virtual machine is to be moved.
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyVm.

Examples

Example A.160 Moving a virtual machine to a storage repository

```
OVM> moveVmToRepository Vm name=MyVm cloneCustomizer=MyVMCloneCustomizer \
  targetRepository=MyRepository
```

See Also

- Section A.52, “create VmCloneCustomizer”
- Section A.39, “create PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.124 refresh

Refreshes configuration information about an object in Oracle VM Manager.

Syntax

`refresh` { `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command reads the configuration information about the object and updates the Oracle VM Manager database repository.

When refreshing the file systems or a file server with non-uniform exports, all of the refresh servers for the file server must be available. If a refresh server is out of commission and you need to perform a refresh, you can remove that refresh server from the file server and add an alternate that has access to the same set of exports on the file server. For more information on uniform and non-uniform exports, please refer to What are Uniform and Non-uniform Exports? in the *Oracle VM Concepts Guide*.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `PhysicalDisk` | `Repository` | `Server` | `StorageArray` }	The object to be refreshed.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`. Note that if the object name contains forward slashes, these must be escaped using an additional forward slash. This is illustrated in the examples for this command.

Examples

Example A.161 Refreshing a file server

```
OVM> refresh FileServer name=MyNFSServer
```

Example A.162 Refreshing a storage array

```
OVM> refresh StorageArray name=MyISCSIServer
```

Example A.163 Refreshing a physical disk

```
OVM> refresh PhysicalDisk id=0004fb000018000035ce16ee4d58dc4d
```

Example A.164 Refreshing a file system

```
OVM> refresh FileSystem name="nfs on 10.172.76.125://mnt//vol1//repo01"
```

Example A.165 Refreshing a storage repository

```
OVM> refresh Repository name=MyRepository
```

See Also

- Section A.125, “refreshAll”
- Section A.126, “refreshStorageLayer”

A.125 refreshAll

Rediscovers all Oracle VM Server instances, file servers, and storage arrays.

Syntax

`refreshAll`

Description

This command rediscovers all Oracle VM Server instances, file servers, and storage arrays.

> **Important**
>
> The **Refresh All** function does not pick up the contents of file systems that have never been refreshed before. Furthermore, it does not refresh repositories that are not already presented on at least one server. It is important to keep this in mind if you have restored a configuration from a backup, since some items may not have been refreshed before at the time that the backup was created.

Options

This command does not take any arguments or provide any options.

Examples

Example A.166 Rediscovering Oracle VM Servers, file servers and storage arrays

```
OVM> refreshAll
```

See Also

- Section A.124, “refresh”
- Section A.126, “refreshStorageLayer”

A.126 refreshStorageLayer

Refreshes the storage visible to an Oracle VM Server.

Syntax

`refreshStorageLayer Server` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command refreshes the storage visible to an Oracle VM Server. This command discovers:

- Exported NFS shares residing on any NFS file servers. The Oracle VM Server must be an admin server or refresh server.
- Presented physical disks on any iSCSI or unmanaged Fibre Channel storage arrays. The Oracle VM Server must be an admin server.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.167 Refreshing Oracle VM Server storage

```
OVM> refreshStorageLayer Server name=MyServer
```

See Also

- Section A.124, “refresh”
- Section A.125, “refreshAll”
- Section A.121, “list”
- Section A.151, “show”

A.127 releaseOwnership

Releases ownership of an Oracle VM Server or storage repository.

Syntax

`releaseOwnership` { `Repository` | `Server` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command releases ownership of an Oracle VM Server or storage repository.

Options

The following table shows the available options for this command.

Option	Description
{ `Repository` | `Server` }	The object for which to release ownership, either an Oracle VM Server or a storage repository.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.168 Releasing ownership of an Oracle VM Server

```
OVM> releaseOwnership Server name=MyServer
```

Example A.169 Releasing ownership of a storage repository

```
OVM> releaseOwnership Repository name=MyRepository
```

See Also

- Section A.159, “takeOwnership”

A.128 removeAccessHost

Removes an access host from an ISCSI server.

Syntax

`removeAccessHost StorageArray` *`instance`* `accessHost=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes an access host from an ISCSI server when using a storage array with *multipath* capability. At least one access host must be set. Multipath is not supported with the generic ISCSI storage array plug-in. This is not applicable to fibre channel storage.

Options

The following table shows the available options for this command.

Option	Description
`accessHost=`*`value`*	The hostname or IP address for the access host. To find the hostname or IP address, use the show command to display information about the storage array.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyISCSIServer`.

Examples

Example A.170 Removing a storage array access host

```
OVM> removeAccessHost StorageArray name=MyISCSIServer accessHost=10.172.76.131
```

See Also

- Section A.3, “addAccessHost”
- Section A.47, “create StorageArray”
- Section A.84, “edit StorageArray”
- Section A.121, “list”
- Section A.151, “show”

A.129 removeAdminServer

Removes an administrative Oracle VM Server from a file server or storage array.

Syntax

`removeAdminServer` { `FileServer` | `StorageArray` } *`instance`* `server=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes an administrative Oracle VM Server from a file server or storage array.

Options

The following table shows the available options for this command.

Option	Description
`server=`*`value`*	The name or ID of the administrative Oracle VM Server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyFileServer`.

Examples

Example A.171 Removing an admin server from a file server

```
OVM> removeAdminServer FileServer name=MyNFSServer server=MyServer
```

See Also

- Section A.4, “addAdminServer”
- Section A.36, “create FileServer”
- Section A.47, “create StorageArray”
- Section A.68, “edit FileServer”
- Section A.84, “edit StorageArray”
- Section A.136, “remove Server”

- Section A.11, “add Server”
- Section A.6, “addRefreshServer”
- Section A.131, “removeRefreshServer”
- Section A.84, “edit StorageArray”

A.130 removePolicyServer

Removes a policy server from a server pool.

Syntax

`removePolicyServer` `ServerPool` *`instance`* `server=`*`value`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes a policy server from a server pool.

Options

The following table shows the available options for this command.

Option	Description
`server=`*`value`*	The name or ID of an Oracle VM Server to remove as a policy server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.172 Removing a policy server from a server pool

```
OVM> removePolicyServer ServerPool name=MyServerPool server=MyServer
```

See Also

- Section A.5, “addPolicyServer”
- Section A.44, “create ServerPoolNetworkPolicy”
- Section A.81, “edit ServerPoolNetworkPolicy”

A.131 removeRefreshServer

Removes an Oracle VM Server that is able to refresh a file server.

Syntax

`removeRefreshServer` `FileServer` *`instance`* `server=`*`value`*

Where *instance* is:

{ `id=`*value* | `name=`*value* }

Description

This command removes a *refresh server* from a file server. The refresh server is an Oracle VM Server that is used to refresh the file systems on an NFS file server. A file server must have at least one refresh server assigned to it.

Options

The following table shows the available options for this command.

Option	Description
`FileServer` *instance*	The name or ID of the file server.
`server=`*value*	The name or ID of the Oracle VM Server to be removed as a refresh server.
{ `id=`*value* | `name=`*value* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.173 Removing a refresh server from a file server

```
OVM> removeRefreshServer FileServer name=MyNFSServer server=MyServer1
```

See Also

- Section A.6, “addRefreshServer”
- Section A.36, “create FileServer”
- Section A.68, “edit FileServer”
- Section A.136, “remove Server”
- Section A.11, “add Server”
- Section A.4, “addAdminServer”
- Section A.129, “removeAdminServer”
- Section A.124, “refresh”

A.132 remove BondPort

Removes a bond port from a network object.

Syntax

`remove BondPort` *instance* `from Network` *instance*

Where *instance* is:

{ `id=value` | `name=value` }

Description

This command removes a bond port from a network object.

Options

The following table shows the available options for this command.

Option	Description
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyBondPort`.

Examples

Example A.174 Removing a bonded port from a network

```
OVM> remove BondPort id=0004fb000020000065822cb7bb9ec296 from Network name=MyVMNetwork
```

See Also

- Section A.34, “create BondPort”
- Section A.64, “edit BondPort”
- Section A.7, “add BondPort”
- Section A.135, “remove Port”
- Section A.10, “add Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.133 remove FileSystem

Removes a file system from an access group.

Syntax

`remove FileSystem` *`instance`* `from AccessGroup` *`instance`*

Where *`instance`* is:

{ `id=value` | `name=value` }

Note that if you need to quote the instance name and it contains forward slashes, you need to escape those forward slashes with additional forward slashes. This is illustrated in the example for this command.

Description

This command removes a file system from an access group.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyFileSystem`.

Examples

Example A.175 Removing a file system from an access group

```
OVM> remove FileSystem name="nfs on 10.172.76.125://mnt//vol2//repo03" from AccessGroup \
  name=MyAccessGroup
```

See Also

- Section A.37, “create FileSystem”
- Section A.70, “edit FileSystem”
- Section A.8, “add FileSystem”
- Section A.32, “create AccessGroup”
- Section A.59, “edit AccessGroup”
- Section A.121, “list”
- Section A.151, “show”
- Section A.57, “delete”

A.134 remove PhysicalDisk

Removes a physical disk from a SAN storage access group.

Syntax

`remove PhysicalDisk` *`instance`* `from AccessGroup` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes a physical disk from a SAN storage access group. Local storage and generic storage plug-ins are not supported with this command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyDisk`.

Examples

Example A.176 Removing physical disk from a SAN storage access group

```
OVM> remove PhysicalDisk id=0004fb00001800007ee6dbda7b4461cb from AccessGroup \
  name='Default access group @ MyISCSIServer'
```

See Also

- Section A.39, “create PhysicalDisk”
- Section A.9, “add PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.135 remove Port

Removes an Ethernet port from a network object.

Syntax

`remove Port` *`instance`* `from` { `BondPort` | `Network` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes an Ethernet port from a network object.

Options

The following table shows the available options for this command.

Option	Description
{ `BondPort` | `Network` }	The network object from which to remove the Ethernet port.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyPort`.

Examples

Example A.177 Removing an Ethernet port from a network

```
OVM> remove Port id=0004fb0000200000be8fa354cb7d98ae from Network name=MyVMNetwork
```

See Also

- Section A.34, “create BondPort”
- Section A.64, “edit BondPort”
- Section A.7, “add BondPort”
- Section A.10, “add Port”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.136 remove Server

Removes an Oracle VM Server from an object.

Syntax

`remove Server` *`instance`* `from` { `AccessGroup` | `CpuCompatibilityGroup` | `Repository` | `ServerPool` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes an Oracle VM Server from either a *CPU compatibility group*, server pool, storage repository or access group.

To remove admin servers from a file server or storage array, use the removeAdminServer command.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `CpuCompatibilityGroup` | `Repository` | `ServerPool` }	The object from which to remove the Oracle VM Server.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.178 Removing an Oracle VM Server from a *CPU compatibility group*

```
OVM> remove Server name=MyServer from CpuCompatibilityGroup name=MyCPUGroup
```

Example A.179 Removing an Oracle VM Server from a server pool

```
OVM> remove Server name=MyServer from ServerPool name=MyServerPool
```

Example A.180 Removing an Oracle VM Server from an access group

```
OVM> remove Server name=MyServer from AccessGroup name=MyAccessGroup
```

Example A.181 Removing (unpresenting) an Oracle VM Server from a storage repository

```
OVM> remove Server name=MyServer from Repository name=MyRepository
```

See Also

- Section A.58, “discoverServer”
- Section A.11, “add Server”
- Section A.78, “edit Server”
- Section A.124, “refresh”
- Section A.156, “start”
- Section A.157, “stop”
- Section A.144, “restart”
- Section A.120, “kill”
- Section A.160, “upgrade”
- Section A.136, “remove Server”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.137 remove ServerPool

Removes a server pool from a storage repository or file system access group.

Syntax

`remove ServerPool` *`instance`* `from` { `AccessGroup` | `Repository` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command unpresents a storage repository from the Oracle VM Servers in a server pool. To unpresent a storage repository to an individual Oracle VM Server, use the remove Server command.

This command also removes a server pool from an access group.

Options

The following table shows the available options for this command.

Option	**Description**
{ `AccessGroup` | `Repository` }	The object from which to remove the server pool.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServerPool`.

Examples

Example A.182 Unpresenting a storage repository from a server pool

```
OVM> remove ServerPool name=MyServerPool from Repository name=MyNFSRepository
```

Example A.183 Removing a server pool from an access group

```
OVM> remove ServerPool name=MyServerPool from AccessGroup name=MyAccessGroup
```

See Also

- Section A.12, “add ServerPool”
- Section A.136, “remove Server”
- Section A.121, “list”
- Section A.151, “show”

A.138 remove StorageInitiator

Removes a storage initiator from an access group for a SAN storage server.

Syntax

`remove StorageInitiator` *`instance`* `from AccessGroup` *`instance`*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command removes a storage initiator from an access group for a SAN storage server.

Options

The following table shows the available options for this command.

Option	Description
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyStorageInitiator.

Examples

Example A.184 Removing a storage initiator

```
OVM> remove StorageInitiator name=iqn.1988-12.com.oracle:d72d82d0817f from AccessGroup \
  name='Default access group @ MyISCSIServer'
```

See Also

- Section A.13, “add StorageInitiator”
- Section A.121, “list”
- Section A.151, “show”

A.139 remove Tag

Removes a tag from an object.

Syntax

remove Tag *instance* from { Server | ServerPool | Vm } *instance*

Where *instance* is:

{ id=*value* | name=*value* }

Description

This command removes a tag used to identify and group objects from an object.

Options

The following table shows the available options for this command.

Option	Description
{ Server | ServerPool | Vm }	The object from which to remove the tag.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyTag`.

Examples

Example A.185 Removing a tag from a server pool

```
OVM> remove Tag name=MyTag from ServerPool name=MyServerPool
```

See Also

- Section A.48, “create Tag”
- Section A.86, “edit Tag”
- Section A.14, “add Tag”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.140 remove VlanInterface

Removes a VLAN interface from a network.

Syntax

`remove VlanInterface` *`instance`* `from Network` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes a VLAN interface from a network.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVlanInterface`.

Examples

Example A.186 Removing a VLAN interface from a network

```
OVM> remove VlanInterface name=MyVLANInterface from Network name=MyNetwork
```

See Also

- Section A.50, “create VlanInterface”
- Section A.15, “add VlanInterface”
- Section A.89, “edit VlanInterface”
- Section A.97, “embeddedCreate”
- Section A.98, “embeddedDelete”
- Section A.99, “embeddedEdit”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.141 remove Vm

Removes a virtual machine from an Oracle VM Server, server pool, or anti affinity group.

Syntax

`remove Vm` *`instance`* `from` { `AntiAffinityGroup` | `Server` | `ServerPool` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes a virtual machine from an Oracle VM Server, server pool, or anti affinity group. The virtual machine cannot be running, and must be stopped before using this command.

In the Oracle VM Manager Web Interface:

- When you remove a virtual machine from an Oracle VM Server, it is moved to the server pool and is available by displaying the **Virtual Machines** perspective at the server pool level in the **Servers and VMs** tab.
- When you remove a virtual machine from a server pool, it is moved to the **Unassigned Virtual Machines** folder in the **Servers and VMs** tab.

Options

The following table shows the available options for this command.

Option	Description
{ `AntiAffinityGroup` | `Server` | `ServerPool` }	The object from which to remove the virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.187 Removing a virtual machine from a server pool

```
OVM> remove Vm name=MyVM from ServerPool name=MyServerPool
```

Example A.188 Removing a virtual machine from an Oracle VM Server

```
OVM> remove Vm name=MyVM from Server name=MyServer
```

Example A.189 Removing a virtual machine from an anti affinity group

```
OVM> remove Vm name=MyVM from AntiAffinityGroup name=MyAAGroup
```

See Also

- Section A.16, “add Vm”
- Section A.121, “list”
- Section A.151, “show”

A.142 remove Vnic

Removes a VNIC from a network.

Syntax

`remove Vnic` *`instance`* `from Network` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command removes a VNIC from a network.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVNIC`.

Examples

Example A.190 Removing a VNIC from a network

```
OVM> remove Vnic name=00:21:f6:00:00:00 from Network name=MyNetwork
```

See Also

- Section A.56, “create Vnic”
- Section A.95, “edit Vnic”

- Section A.17, “add Vnic”
- Section A.150, “setVnicMacAddrRange”
- Section A.113, “getVnicMacAddrRange”
- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.143 resize

Resizes a physical or virtual disk.

Syntax

`resize` { `PhysicalDisk` | `VirtualDisk` } *`instance`* `size=`*`value`* `sparse=` { `Yes` | `No` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command resizes a physical or virtual disk. The `sparse` option is only available when resizing a virtual disk.

Options

The following table shows the available options for this command.

Option	Description
`size=`*`value`*	The size of the physical or virtual disk in GiB.
`sparse=` { `Yes` | `No` }	Whether to create a sparse or non-sparse virtual disk. This option is only available with the `resize VirtualDisk` command.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVirtualDisk`.

Examples

Example A.191 Resizing a virtual disk

```
OVM> resize VirtualDisk name=MyVMDisk size=200 sparse=Yes
```

Example A.192 Resizing a physical disk

```
OVM> resize PhysicalDisk name=MyVMDisk size=200
```

See Also

- Section A.39, “create PhysicalDisk”

- Section A.9, “add PhysicalDisk”
- Section A.74, “edit PhysicalDisk”
- Section A.134, “remove PhysicalDisk”
- Section A.118, “importVirtualDisk”
- Section A.49, “create VirtualDisk”
- Section A.88, “edit VirtualDisk”
- Section A.57, “delete”
- Section A.124, “refresh”
- Section A.121, “list”
- Section A.151, “show”
- Section A.103, “getEvents”

A.144 restart

Restarting an Oracle VM Server or virtual machine.

Syntax

`restart` { `Server` | `Vm` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command restarts an Oracle VM Server or virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `Server` | `Vm` }	The object to restart, either an Oracle VM Server or a virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.193 Restarting an Oracle VM Server

```
OVM> restart Server name=MyServer
```

Example A.194 Restarting a virtual machine

```
OVM> restart Vm name=MyVM
```

See Also

- Section A.156, “start”
- Section A.157, “stop”
- Section A.120, “kill”

A.145 resume

Resumes a suspended virtual machine.

Syntax

`resume` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command resumes a suspended virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.195 Resuming a virtual machine

```
OVM> resume Vm name=MyVM
```

See Also

- Section A.158, “suspend”
- Section A.121, “list”
- Section A.151, “show”

A.146 sendVmMessage

Sends a key/value pair message to a running virtual machine.

Syntax

`sendVmMessage` `Vm` *`instance`* `key=`*`value`* `message=`*`value`* `log=` { `Yes` | `No` }

Where *`instance`* is:

{ `id=value` | `name=value` }

Description

This command sends a key/value pair message to a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
`key=value`	The message key.
`message=value`	The message content.
`log=` { `Yes` | `No` }	Whether to log the message.
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.196 Sending a message to a virtual machine

```
OVM> sendVmMessage Vm name=MyVM key=com.oracle.linux.network.device.0 message=eth0 log=No
```

See Also

- Section A.112, "getVmSentMessages"
- Section A.111, "getVmReceivedMessages"
- Section A.22, "clearVmRcvdMessage"
- Section A.23, "clearVmSentMessage"
- Section A.21, "clearVmAllSentMessages"
- Section A.20, "clearVmAllRcvdMessages"

A.147 set

Sets configuration options for the CLI.

Syntax

`set` { `CommandMode=` { `Asynchronous` | `Synchronous` } | `CommandTimeout=` *`value`* | `EndlineChars=` { `CRLF` | `CR` | `LF` } | `OutputMode=` { `Verbose` | `Sparse` | `Xml` } }

Description

This command sets configuration options for the CLI.

Options

The following table shows the available options for this command.

Option	Description
`CommandMode=` { `Asynchronous` | `Synchronous` }	Sets whether the CLI should be run in synchronous or asynchronous mode.
`CommandTimeout=` *`value`*	Sets when the CLI will timeout. *`value`* can be an integer between 1 and 43200.
`EndlineChars=` { `CRLF` | `CR` | `LF` }	Sets the end of line character to use for your SSH client, for example, if your SSH client adds a line feed (double spacing) to the end of a line, you can set the end of line characters to `CR`. `CRLF` is used on Microsoft Windows systems, `CR` is used on early Apple systems, and `LF` is used on Linux, and Unix-like systems such a modern Apple systems.
`OutputMode=` { `Verbose` | `Sparse` | `Xml` }	Sets the output mode for command results. `Verbose` includes the command, status, time and time zone. `Sparse` returns just the results without the header files provided by `Verbose`. `Xml` returns the results in XML format.

Examples

Example A.197 Setting end of line characters

```
OVM> set EndlineChars=LF
```

Example A.198 Setting output mode to XML

```
OVM> set OutputMode=Xml
```

A.148 setMaintenanceMode

Sets maintenance mode on an Oracle VM Server.

Syntax

`setMaintenanceMode` `Server` *`instance`* `mode=` { `on` | `off` }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command sets whether an Oracle VM Server is in maintenance mode in order to perform software updates.

Options

The following table shows the available options for this command.

Option	Description
`mode=` { `on` | `off` }	Whether to place the Oracle VM Server into maintenance mode.

Option	Description
{ id=*value* | name=*value* }	The instance of the object using either the id or name option, for example name=MyServer.

Examples

Example A.199 Placing an Oracle VM Server into maintenance mode

```
OVM> setMaintenanceMode Server name=MyServer mode=on
```

See Also

- Section A.160, “upgrade”
- Section A.121, “list”
- Section A.151, “show”

A.149 setStatsConfig

Sets the configuration for the statistics displayed in Oracle VM Manager.

Syntax

setStatsConfig samplingInterval=*value* holdTime=*value*

Description

This command sets the configuration for the statistics displayed in Oracle VM Manager.

Options

The following table shows the available options for this command.

Option	Description
samplingInterval=*value*	The number of seconds between which statistics should be recorded. *value* may be an integer between 20 and 31536000.
holdTime=*value*	The number of minutes to retain the statistics. *value* may be an integer between 15 and 4320.

Examples

Example A.200 Setting the Oracle VM Manager statistics configuration

```
OVM> setStatsConfig samplingInterval=60 holdTime=15
```

See Also

- Section A.107, “getStatsConfig”

A.150 setVnicMacAddrRange

Sets the range of MAC addresses that are available to VNICs.

Syntax

`setVnicMacAddrRange` [`oui=`*`value`*] [`start=`*`value`*] [`end=`*`value`*]

Description

This command sets the range of MAC addresses that can be used when creating a VNIC. To display the MAC address range, use the getVnicMacAddrRange command.

Options

The following table shows the available options for this command.

Option	Description
`oui=`*`value`*	The OUI (Organizationally Unique Identifier) is used as the first three octets of the MAC address. The MAC address is created by combining the OUI as the first three octets with the randomly selected second three octets in the range of start to end values specified, inclusive. Therefore, the value specified here should represent the first three octets that make up the MAC address. The default OUI is 00:21:f6 and is owned by Oracle. Changing the OUI can result in an overlap of MAC addresses on your network causing MAC address spooking, network conflicts and unexpected network behavior. It is recommended that you do not change this value.
`start=`*`value`*	The parameter used to specify the first possible value for the second three octets that form the MAC address. The default start value is 00:00:00.
`end=`*`value`*	The parameter used to specify the final possible value for the second three octets that form the MAC address. The default start value is FF:FF:FF.

Examples

Example A.201 Setting the VNIC MAC Address Range

```
OVM> setVnicMacAddrRange oui=00:21:f6 start=00:00:00 end=FF:FF:FF
```

See Also

- Section A.113, “getVnicMacAddrRange”
- Section A.56, “create Vnic”
- Section A.95, “edit Vnic”
- Section A.17, “add Vnic”
- Section A.142, “remove Vnic”

- Section A.57, “delete”
- Section A.121, “list”
- Section A.151, “show”

A.151 show

Shows information about an object.

Syntax

`show` { `AccessGroup` | `AntiAffinityGroup` | `Assembly` | `AssemblyVirtualDisk` | `AssemblyVm` | `BondPort` | `ControlDomain` | `Cpu` | `CpuCompatibilityGroup` | `FileServer` | `FileServerPlugin` | `FileSystem` | `Job` | `Manager` | `Network` | `PeriodicTask` | `PhysicalDisk` | `Port` | `Repository` | `RepositoryExport` | `Server` | `ServerController` | `ServerPool` | `ServerPoolNetworkPolicy` | `ServerUpdateGroup` | `ServerUpdateRepository` | `StorageArray` | `StorageArrayPlugin` | `StorageInitiator` | `Tag` | `VirtualCdrom` | `VirtualDisk` | `VlanInterface` | `Vm` | `VmCloneCustomizer` | `VmCloneNetworkMapping` | `VmCloneStorageMapping` | `VmDiskMapping` | `Vnic` | `VolumeGroup` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command shows information about an object. Use the `list` command to find all instances of an object type, then use the `show` command to show more detailed information about the object.

A Job object does not have a `name` attribute, only an `id` attribute. The `show Job name=`*`value`* command is the same as entering `show Job id=`*`value`*. You can use these two options interchangeably. Any `name` attributes are automatically converted to `id`s.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `AntiAffinityGroup` | `Assembly` | `AssemblyVirtualDisk` | `AssemblyVm` | `BondPort` | `ControlDomain` | `Cpu` | `CpuCompatibilityGroup` | `FileServer` | `FileServerPlugin` | `FileSystem` | `Job` | `Manager` | `Network` | `PeriodicTask` | `PhysicalDisk` | `Port` | `Repository` | `RepositoryExport` | `Server` | `ServerController` | `ServerPool` | `ServerPoolNetworkPolicy` | `ServerUpdateGroup` | `ServerUpdateRepository` | `StorageArray` |	The object about which to show information.

Option	Description
`StorageArrayPlugin` | `StorageInitiator` | `Tag` | `VirtualCdrom` | `VirtualDisk` | `VlanInterface` | `Vm` | `VmCloneCustomizer` | `VmCloneNetworkMapping` | `VmCloneStorageMapping` | `VmDiskMapping` | `Vnic` | `VolumeGroup` }	
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.202 Showing details about an Oracle VM Server

```
OVM> show Server name=MyServer
```

Example A.203 Showing details about a virtual machine

```
OVM> show Vm name=MyVM
```

Example A.204 Showing details about a network

```
OVM> show Network id=0004fb0010ff705
```

Example A.205 Showing details about a VNIC

```
OVM> show Vnic name=00:21:f6:00:00:0b
```

Using the `show Vnic` command can provide you with information such as the virtual machine that is using a particular VNIC, as well as the IP addresses that are configured for that VNIC.

Note

IP addresses are only displayed for VNICs attached to virtual machines that have been properly set up with the Oracle VM Guest Additions packages. See the *Oracle VM Administrator's Guide* for information on using the Oracle VM Guest Additions packages.

See Also

- Section A.121, “list”

A.152 showallcustomcmds

Provides a list of all commands and the objects that they relate to.

Syntax

`showallcustomcmds`

Description

This command provides a list of all commands along with the objects that the commands relate to.

Options

This command does not take any arguments or provide any options.

Examples

Example A.206 Showing all custom commands

```
OVM> showallcustomcmds
```

See Also

- Section A.114, “help”
- Section A.153, “showcustomcmds”

A.153 showcustomcmds

Shows available custom commands for an object type.

Syntax

`showcustomcmds` {{ `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `Job` | `PhysicalDisk` | `Repository` | `StorageArray` | `Server` | `ServerPool` | `VirtualDisk` | `Vm` | `VolumeGroup` }}

Description

This command shows the available custom commands specific to an object. Use the `showobjtypes` command to find all object types, then use the `showcustomcmds` command to show associated commands.

Note

Not all object types have custom commands associated. For example, the **YumConfig** object type does not have any custom commands.

Options

The following table shows the available options for this command.

Option	Description
{ `AccessGroup` | `Assembly` | `FileServer` | `FileSystem` | `Job` | `PhysicalDisk` | `Repository` | `StorageArray` | `Server` | `ServerPool` | `VirtualDisk` | `Vm` | `VolumeGroup` }	The object type for which to list the custom commands.

Examples

Example A.207 Showing custom commands for an Oracle VM Server

```
OVM> showcustomcmds Server
```

Example A.208 Showing custom commands for a virtual machine

```
OVM> showcustomcmds VM
```

Example A.209 Showing custom commands for a repository

```
OVM> showcustomcmds Repository
```

See Also

- Section A.114, “help”
- Section A.152, “showallcustomcmds”
- Section A.154, “showobjtypes”

A.154 showobjtypes

Provides a list of all object types.

Syntax

```
showobjtypes
```

Description

This command provides a list of all object types. This command is useful to assist in determining which object types can be acted upon.

Options

This command does not take any arguments or provide any options.

Examples

Example A.210 Showing all object types

```
OVM> showobjtypes
```

See Also

- Section A.103, “getEvents”
- Section A.104, “getEventsForObject”
- Section A.114, “help”
- Section A.153, “showcustomcmds”

A.155 showversion

Shows the version number of the CLI/Oracle VM Manager.

Syntax

```
showversion
```

Description

This command shows the version number of the CLI/Oracle VM Manager.

Options

This command does not take any arguments or provide any options.

Examples

Example A.211 Showing the CLI/Oracle VM Manager version number

```
OVM> showversion
```

See Also

- Section A.114, “help”

A.156 start

Starts an Oracle VM Server or virtual machine.

Syntax

`start` { `Server` | `Vm` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command starts an Oracle VM Server or virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `Server` | `Vm` }	The object to start, either an Oracle VM Server or a virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.212 Starting an Oracle VM Server

```
OVM> start Server name=MyServer
```

Example A.213 Starting a virtual machine

```
OVM> start Vm name=MyVM
```

See Also

- Section A.144, “restart”
- Section A.157, “stop”
- Section A.120, “kill”

A.157 stop

Stops an Oracle VM Server or virtual machine.

Syntax

`stop` { `Server` | `Vm` } *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command stops an Oracle VM Server or virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `Server` | `Vm` }	The object to stop, either an Oracle VM Server or a virtual machine.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.214 Stopping an Oracle VM Server

```
OVM> stop Server name=MyServer
```

Example A.215 Stopping a virtual machine

```
OVM> stop Vm name=MyVM
```

See Also

- Section A.156, “start”
- Section A.144, “restart”
- Section A.120, “kill”

A.158 suspend

Suspends a running virtual machine.

Syntax

`suspend` `Vm` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command suspends a running virtual machine.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyVM`.

Examples

Example A.216 Suspending a virtual machine

```
OVM> suspend Vm name=MyVM
```

See Also

- Section A.145, “resume”
- Section A.121, “list”
- Section A.151, “show”

A.159 takeOwnership

Take ownership of an Oracle VM Server or storage repository.

Syntax

`takeOwnership` { `Repository` *`instance`* [`serverpool=`*`value`*] | `Server` *`instance`* `password=`*`value`* }

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command takes ownership of an Oracle VM Server or storage repository. After taking ownership of a repository, you should refresh it using the refresh command. If a server is only partially discovered, in the sense that it is already under the ownership of another Oracle VM Manager instance, and ownership is subsequently released, you may need to rediscover the server before you are able to take ownership of it in the current Oracle VM Manager instance.

Options

The following table shows the available options for this command.

Option	Description
{ `Repository` *`instance`* [`serverpool=`*`value`*] | `Server` *`instance`* `password=`*`value`* }	The object for which to take ownership, either an Oracle VM Server or a storage repository. When taking ownership of a storage repository that uses an OCFS2-based file system, you should enter the server pool to which the repository is provided using the `serverpool` option. When taking ownership of an Oracle VM Server, you should provide the password for the Oracle VM Agent using the `password` option.
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.217 Taking ownership of an Oracle VM Server

```
OVM> takeOwnership Server name=MyServer password=********
```

Example A.218 Taking ownership of a storage repository

```
OVM> takeOwnership Repository name=MyRepository serverpool=MyServerPool
```

See Also

- Section A.127, “releaseOwnership”
- Section A.124, “refresh”

A.160 upgrade

Upgrades an Oracle VM Server.

Syntax

`upgrade` `Server` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command updates or upgrades an Oracle VM Server using a server update repository. The repository used for the upgrade is set using the create ServerUpdateRepository command. This command places the

Oracle VM Server into maintenance mode, checks for any updates in the server update repository, installs any updates, restarts the Oracle VM Server, then takes it out of maintenance mode and returns it to the server pool as a fully functioning member of the pool.

You can check if an Oracle VM Server has an update available using the checkUpToDate command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=`*`value`* | `name=`*`value`* }	The instance of the object using either the `id` or `name` option, for example `name=MyServer`.

Examples

Example A.219 Upgrading an Oracle VM Server

```
OVM> upgrade Server name=MyServer
```

See Also

- Section A.45, “create ServerUpdateGroup”
- Section A.82, “edit ServerUpdateGroup”
- Section A.46, “create ServerUpdateRepository”
- Section A.83, “edit ServerUpdateRepository”
- Section A.19, “checkUpToDate”
- Section A.148, “setMaintenanceMode”
- Section A.121, “list”
- Section A.151, “show”

A.161 validate

Validates a storage array.

Syntax

`validate` `StorageArray` *`instance`*

Where *`instance`* is:

{ `id=`*`value`* | `name=`*`value`* }

Description

This command validates a storage array using the storage array plug-in. Validation is required after the storage array is discovered and after modification of storage array attributes. At least one administrative Oracle VM Server must be configured before using this command.

Options

The following table shows the available options for this command.

Option	Description
{ `id=value` | `name=value` }	The instance of the object using either the `id` or `name` option, for example `name=MyStorageArray`.

Examples

Example A.220 Validating a storage array

```
OVM> validate StorageArray name=MyISCSIServer
```

See Also

- Section A.124, “refresh”
- Section A.125, “refreshAll”
- Section A.121, “list”
- Section A.151, “show”

Glossary

A

admin server
: An Oracle VM Server dedicated to performing administrative functions on storage servers such as creating a new LUN or extending a file system. The server must be capable of logging into a storage array or file server as an admin user. The administrative functions available to the server are defined by the Oracle VM Storage Connect plug-in.

assembly
: An infrastructure template containing a configuration of multiple virtual machines with their virtual disks, and the inter-connectivity between them. Assemblies can be created as a set of .ovf (Open Virtualization Format) and .img (disk image) files, or may all be contained in a single .ova (Open Virtualization Format Archive) file.

B

bonding
: Bonding is a Linux OS feature that provides a method for aggregating several ports into a single bonded interface, to provide load balancing or redundancy. When you discover an Oracle VM Server, the bonded interface is shown as containing a single port.

Network bonding refers to the combination of network interfaces on one host for redundancy and/or increased throughput. Redundancy is the key factor: You want to protect your virtualized environment from loss of service due to failure of a single physical link.

In Oracle VM, there are three modes of network bonding:

- Active - Passive: One Network Interface Card (NIC) is active while the other NIC is standby. If the active NIC goes down, the other NIC becomes active.
- Dynamic Link Aggregation: All NICs act as one NIC and the network traffic flows through all interfaces concurrently, which results in a higher throughput. With this mode, your network administrator must create LACP (Link Aggregation Control Protocol) bonding on the network switch(es).
- Adaptive Load Balancing: The network traffic is equally balanced over the NICs of the bond. This mode does not require any special configuration on connected network switch(es), However, this mode does not support using VLAN with bridges. If using this mode for your bonded interfaces in any network, you cannot use VLANs if this network is configured with the Virtual Machine network channel.

C

clone
: The action or result of making an exact copy of an object. The object may be a virtual machine, virtual machine template, ISO file, or virtual disk. Cloning is analogous to copying and maintains the integrity of the original object, while creating a new object based on the original. A clone customizer may be used to define cloning options to specify details of where the object components may reside when cloned, such as in a different storage repository.

D

discover
: The process of adding systems as objects within Oracle VM Manager is known as *discovery*. When you add Oracle VM Servers and storage to your Oracle VM environment, Oracle VM Manager uses the information

provided to connect to the resource and perform verification. During this process, information is usually exchanged between the server and the manager. In the case of an Oracle VM Server, Oracle VM Manager obtains information about the server, its network connectivity and any storage that is already attached to the server. Depending on your hardware and networking configuration, external storage may be automatically detected during discovery of Oracle VM Servers. This is always the case with local OCFS2 storage on an Oracle VM Server.

While storage can be automatically discovered during the process of discovering Oracle VM Servers, you may need to perform storage discovery for resources that are not already attached to any of your Oracle VM Servers. It is important that storage is configured outside of the Oracle VM environment prior to discovery. Depending on the storage type, you can perform different storage discovery operations from within Oracle VM Manager.

E

events

Events are used to register status information of "objects" within Oracle VM Manager for future reference or to make problems easier to trace back. Events are often, though not always, related to *jobs* that are initiated within Oracle VM Manager. For instance, when a job fails, an event is generated. Events can also be triggered through changes in the environment such as server crashes or storage disconnects. Therefore, events are used to alert you to potential problems that may need your attention.

Events are categorized by severity. Most events will be informational, but they can also be warnings or errors. If an event has an error level severity, you need to acknowledge the error event to clear the error and to perform further operations on the object that generated the error.

See Also: *jobs*

G

guest

A guest operating system that runs within a domain in Oracle VM Server. A guest may be paravirtualized or hardware virtualized. Multiple guests can run on the same Oracle VM Server.

H

host computer

The physical computer on which the software is installed. Typically used to refer to either the computer on which Oracle VM Server or Oracle VM Manager is running.

J

jobs

Jobs are a sequential operations that take place through Oracle VM Manager, such as server discovery, presenting a repository and creating a VM. Jobs are assigned a status that is refreshed according to their progress. A history of all jobs in the environment is stored within Oracle VM Manager.

Since jobs are sequential and sometimes take time to complete, tracking the status of a job allows you to understand what actions the system is currently performing, and which actions are queued to run in sequence after the current job has completed. Jobs also allow you to access system messages that may be useful to debug the failure of an operation.

Most jobs tend to generate events that each have a different severity level.

See Also: *events*

L

local storage

Local storage consists of hard disks installed locally in an Oracle VM Server. Local storage is often not appropriate for production environments, because it prevents or sharply constrains the ability of a virtual machine to run anywhere in the server pool in the event of the failure of the Oracle VM server, which owns the local storage.

M

migrate

The act of moving a virtual machine from one Oracle VM Server to another, or to the Unassigned Virtual Machines folder. Migration can be performed on either a running or a stopped virtual machine.

move

The act of moving an object from one location to another. This may be moving a stopped virtual machine from one Oracle VM Server to another, moving a virtual machine template from one storage repository to another, or moving an ISO file or virtual disk to another storage location.

multipath

The technique of creating more than one physical path between the server CPU and its storage devices. It results in better fault tolerance and performance enhancement. Oracle VM supports multipath I/O out of the box. Oracle VM Servers are installed with multipathing enabled because it is a requirement for SAN disks to be discovered by Oracle VM Manager

O

Oracle VM Server

A self-contained virtualization environment designed to provide a lightweight, secure, server-based platform for running virtual machines. The Oracle VM Server comprises a hypervisor and a privileged domain (called dom0) that allow multiple domains or virtual machines (that is, Linux, Solaris, Windows, and so on) to run on one physical machine. Includes Oracle VM Agent to enable communication with Oracle VM Manager.

The Oracle VM Server for x86 incorporates an open source Xen hypervisor component, which has been customized and optimized to integrate into the larger, Oracle - developed virtualization server. The Oracle VM Server for x86 is also responsible for access and security management and generally acts as the server administrative entity, because the hypervisor's role is limited.

On Oracle VM Server for SPARC systems, the SPARC hypervisor is built into the SPARC firmware and is generally referred to as the Logical Domains Manager (LDOM). As with the Xen hypervisor, each virtual machine is securely executed on a single computer and runs its own guest Oracle Solaris operating system

R

refresh server

An Oracle VM Server dedicated to handling file system refreshes on behalf of a server pool. A refresh server temporarily mounts file systems on an NFS file server during the refresh operation. The server must be granted full data access in order to perform the refresh. For each NFS file server, at least one Oracle VM Server from each server pool accessing the file server must be assigned as a refresh server.

S

server processor compatibility group

A server processor compatibility group is a group of Oracle VM Servers with compatible processors, or CPUs sharing the same processor family and model number. These groups are created to ensure that a virtual machine running on one Oracle VM Server can safely be migrated and continue to run on another Oracle VM Server. Oracle VM Manager automatically creates processor compatibility groups as it discovers servers that have different processor types.

Using Oracle VM Manager you can create custom compatibility groups to improve your ability to do smooth migrations and to group servers according to your own requirements.

Index