

Oracle VM 3: Oracle VM Templates Automated Virtual Machine Provisioning

ORACLE WHITE PAPER | MARCH 2016



Table of Contents	
Introduction	1
Concepts: Oracle VM Guest Additions	1
Messaging channel between Guest and Oracle VM Server	2
Send/Receive Messages between VM and Oracle VM Manager	3
Message Handling inside the Guest	6
Under the hood: Oracle VM Template Configuration	7
Use case: Automated Virtual Machine Provisioning	8
Step by step: Creating an Oracle VM Template with Guest Additions	8
Local Configuration via Virtual Machine Console	12
Remote Configuration via Oracle VM CLI	13
Automated Configuration via Expect	14
Conclusion	16
Appendix (A): Testing of Template Configuration Scripts	17

Introduction

Oracle VM Templates provide an innovative approach to deploying a fully configured software stack by offering pre-installed and pre-configured software images. Use of Oracle VM Templates eliminates the installation and configuration costs, and reduces the ongoing maintenance costs helping organizations achieve faster time to market and lower cost of operations. Oracle VM Templates are part of many key Oracle products available for download, including Oracle Linux, Oracle Solaris, Oracle Database, Fusion Middleware, and many more. Simply download an Oracle VM Template from [Oracle Software Delivery Cloud](#), import it into Oracle VM Manager and then deploy the Template as a virtual machine in order to use the pre-configured software.

[Oracle VM Guest Additions](#) were introduced with Oracle VM 3 that allow the guest software to pass information back and forth through Oracle VM Manager to the virtual machine, and thus provide direct integration between guest software and the virtualization layer, to assist in orchestration of complex, multi-VM deployments.

The rest of this technical white paper focuses on how to automate virtual machine provisioning based on Oracle VM Templates with the Oracle VM Guest Additions. For information on the broader context of Oracle VM Templates, their benefits, and how they are deployed, customized, and used from Oracle VM Manager, refer to the “[Oracle VM Enabling Rapid Migration to Private Cloud](#)” white paper, and visit oracle.com/virtualization for more information about Oracle VM.

Concepts: Oracle VM Guest Additions

Oracle VM Guest Additions is a set of packages that can be installed on the guest operating system of a virtual machine running in the Oracle VM environment. These packages provide the tools to allow bidirectional communication directly between Oracle VM Manager and the operating system running within the virtual machine. This is a powerful tool that provides administrators fine-grained control over the configuration and behavior of components running within the virtual machine directly from Oracle VM Manager.

Features of the Oracle VM Guest Additions include the option to send messages directly to a virtual machine from Oracle VM Manager to trigger programmed events, ability to query a virtual machine from Oracle VM Manager to obtain information, such as the IP address, and the ability to use the template configuration facility to automatically configure virtual machines as they are first started.

These Guest Additions are available for Oracle Linux 5, 6 and 7 from [Oracle's Public YUM repository](#) and can be installed in the guest with the following command:

```
# yum install ovmd xenstoreprovider python-simplejson ovm-template-config
```

This installs the basic necessary packages to support the Oracle VM Guest Additions.

- **ovmd** is a daemon that handles configuration and re-configuration events and provides a mechanism to send/receive messages between the virtual machine and Oracle VM Manager.
- **xenstoreprovider** is an information storage space shared between domains. It looks for specific messages (key-value pairs or events) and passes those to ovmd, or the other way around.
- **python-simplejson** is a simple, fast, extensible JSON encoder/decoder for Python.
- **ovm-template-config** is a collection of OS configuration system scripts used to (re)configure an Oracle VM template when booted up the first time.
- **libovmapi** is the library which communicates with the ovmap kernel infrastructure. This package will be automatically installed because it is a dependency.
- **libovmapi-devel** is an optional package to be installed when creating additional extensions to ovmd.

There is an extra kernel module required to make this work, the **ovmapi kernel module** that provides the ability to communicate messages back and forth between the Oracle VM Server and the VM and as such between Oracle VM Manager and the VM. Since UEK2 (2.6.39) this kernel module is shipped with the kernel.

Next to these basic packages there are also additional Oracle VM Template configuration packages available for configuring the network, system, etc. For more detailed information, see further this white paper or refer to the documentation on [Oracle VM Utilities](#).

Messaging channel between Guest and Oracle VM Server

The Oracle VM Guest Additions daemon, ovmd, facilitates a bi-directional messaging channel between Oracle VM Manager and the guest. It allows first-boot installation configuration, and is capable of sending and receiving messages consisting of key-value pairs.

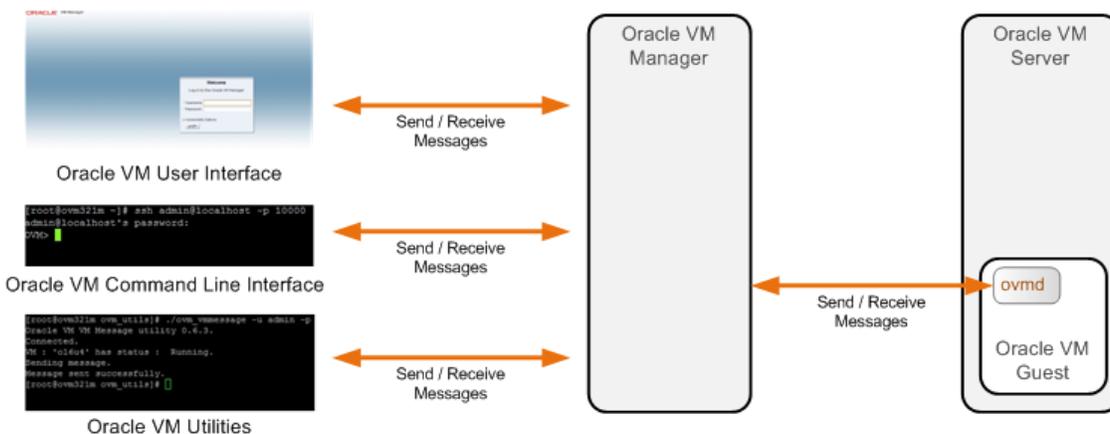


Figure 1. Messaging Channel between Guest and Oracle VM Manager

Sending or receiving messages via Oracle VM Manager can be done in several ways, by using the Oracle VM Manager User Interface (UI), the Oracle VM Command Line Interface (CLI) or the Oracle VM Utilities (ovm_vmmessage utility). Inside the guest, ovmd is responsible for sending or receiving messages.

Send/Receive Messages between VM and Oracle VM Manager

To send a message via the Oracle VM Manager User Interface (UI), select one or more virtual machines, an Oracle VM Server or a server pool, and select the Send VM Messages operation. In the dialog box, select or deselect the virtual machines to use for the send message operation. Each message sent to a virtual machine is contained within its own job. If you send multiple messages to multiple virtual machines, each one has its own job, so 10 messages to 100 virtual machines produces 1,000 jobs.

- **Step 1:** Select your VM(s), right click and select “Send VM Messages...”

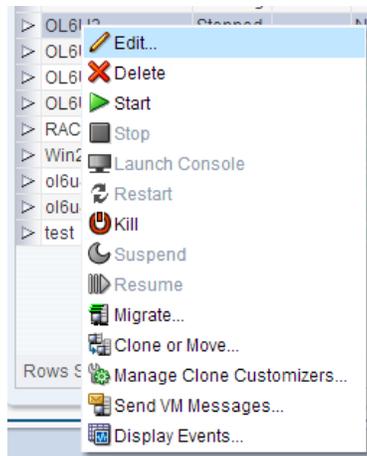


Figure 2. Send VM Messages



- **Step 2:** Under the Messages tab, click on the “+” icon to create a new VM Message

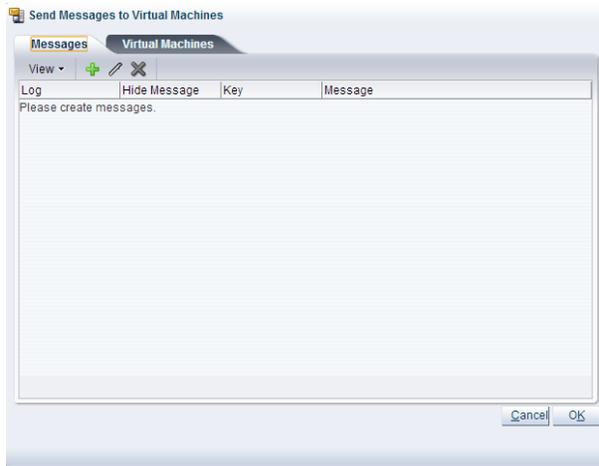


Figure 3. Create a VM Message

- **Step 3:** Fill in a key-value, and click “OK” to send the VM Message

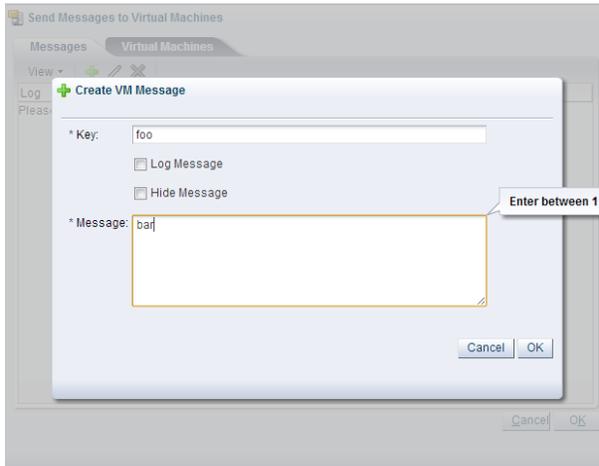


Figure 4. Send a VM Message

Received messages are displayed as events in the Oracle VM Manager UI. To retrieve a message from a virtual machine you can execute following steps:

- **Step 1:** Select your VM, right click and select "Display Events..."

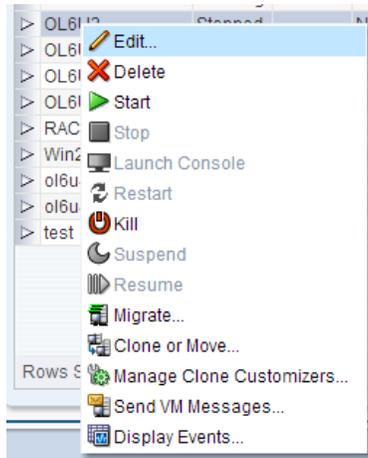


Figure 5. Display Events

- **Step 2:** Look for events with "Virtual Machine API Incoming Message" as summary and check the details of this event for the key-value information.

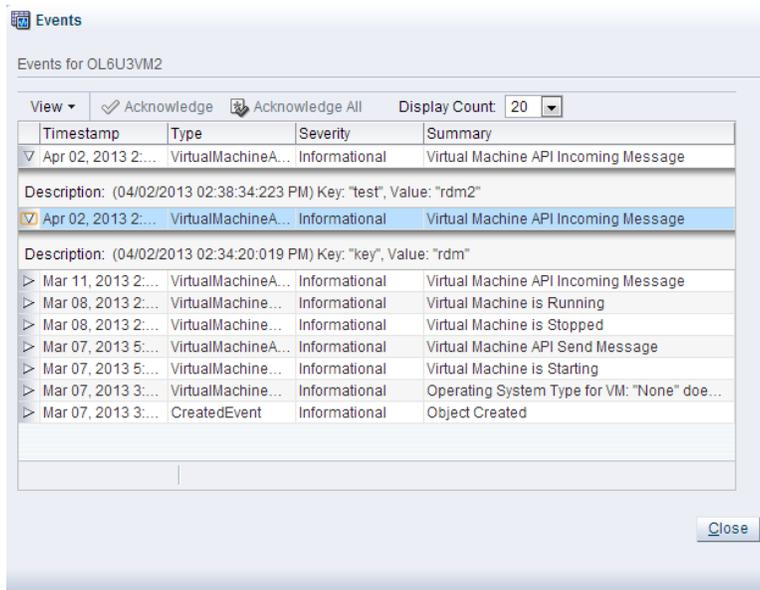


Figure 6. Display Event Information

A second option to send/receive messages is by using the [Oracle VM Command Line Interface \(CLI\)](#). After logging in to the Oracle VM Manager, start the CLI and send a message:

```
# ssh admin@localhost -p 10000
admin@localhost's password: <-- (admin password for the Oracle VM Manager)
OVM> sendVmMessage Vm name=ol6u4 key=foo message=bar log=no
Command: sendVmMessage Vm name=ol6u4 key=foo message=bar log=no
Status: Success
Time: 2013-04-03 09:04:29,890 PST
```

In this example a key-value pair of foo=bar is send to the virtual machine. Retrieving messages can be done by using the getEvents command. Note: at the time of writing the latest available Oracle VM release 3.2 does not have the option to see the content (key-value) of the retrieved message, only that it has been send.

```
# ssh admin@localhost -p 10000
admin@localhost's password:
OVM> getEvents Vm name=ol6u4 type=All amount=2
Status: Success
Time: 2013-04-02 18:42:35,948 CEST
Data:
  id:1364920286848      time:Apr 02, 2013 6:31:26 pm
type:VirtualMachineApiIncomingEvent  severity:Informational  summary:Virtual Machine
API Incoming Message
  id:1364914782589      time:Apr 02, 2013 4:59:42 pm
type:VirtualMachineApiOutgoingEvent  severity:Informational  summary:Virtual Machine
API Send Message
```

As a third option to send/receive messages the [Oracle VM Utilities](#) can be used. After logging in to the Oracle VM Manager, go to /u01/app/oracle/ovm-manager-3/ovm_utils/ and execute following command to send a message:

```
# ./ovm_vmmessage -u admin -p ##### -h localhost -v ol6u4 -k foo -V bar
Oracle VM VM Message utility 0.6.3.
Connected.
VM : 'ol6u4' has status : Running.
Sending message.
Message sent successfully.
```

To retrieve a message use ovm_vmmessage to query (-q option) the value of a key. ovm_vmmessage will also return "when" this key was set inside the virtual machine.

```
# ./ovm_vmmessage -u admin -p ##### -h localhost -v ol6u4 -q com.oracle.linux.root-
password
Oracle VM VM Message utility 0.6.3.
Connected.
VM : 'ol6u4' has status : Running.
Querying for key 'com.oracle.linux.root-password'.
Query successful.
Query for Key : 'com.oracle.linux.root-password' returned value 'password123'.
Key set 225 minutes ago.
```

Message Handling inside the Guest

Inside the guest, the **ovmd** executable can be used to send/receive messages. ovmd has the following options:

```
# ovmd -l          lists all currently set key/value pairs
# ovmd -p key=value sets a key/value pair inside the VM
# ovmd -g key      gets a value from inside the VM
# ovmd -r key      removes a key out of the current cache
# ovmd -x          deletes the key/value values currently set in the cache
```

An example:

```
# ovmd -p foo=bar      <-- (sets a key "foo" with the value "bar")
# ovmd -l              <-- (lists all keys)
{"foo":"bar"}         <-- (there is only one key/value pair)
# ovmd -g foo         <-- (gets the value of key "foo")
bar
# ovmd -r foo         <-- (removes the key "foo")
# ovmd -l              <-- (lists all keys)
#                      <-- (there are no more key/value pairs)
```

With these simple tools it's possible to set up a model, to send messages from an application outside of a virtual machine to a virtual machine through the Oracle VM Guest Additions and also to send messages from an application inside a virtual machine back. This can be done by writing a daemon process that runs and queries for values, or just by doing it manually. A recommendation would be to create a naming convention for this product. For instance, for the Oracle VM Template configuration `com.oracle.linux.[values]` is used. Something similar could be considered or just something like `[application].[key]`. The maximum size of the total message is 8Kb.

Under the hood: Oracle VM Template Configuration

As earlier mentioned `ovmd` is a utility (daemon) that handles configuration and re-configuration events, and provides a mechanism to send/receive messages between a virtual machine and Oracle VM Manager. By enabling the initial-configuration option in the Template, this utility is used to perform first-boot installation configuration either locally from the virtual machine console or remotely through the messaging interface provided by this utility.

During startup of the VM `ovmd -s` configure will be executed which waits till all "required" parameters are received and then all the configuration scripts are executed. By default in the scripts there is only one parameter that is "required" which is the root-password to configure the system root password, all other parameters are optional. Required parameters need to be send at the end of the configuration because once they are received the actual configuration will be executed. Optional parameters send afterwards will be ignored.

To verify the root-password is a required parameter, the following command can be used:

```
# ovm-template-config --human-readable --enumerate --script authentication configure
[('90',
  'authentication',
  [{u'description': u'System root password.',
    u'key': u'com.oracle.linux.root-password',
    u'password': True,
    u'required': True}]]]
```

and to verify this is the only required parameter (by default):

```
# grep "required" /etc/template.d/scripts/*      <--(location of configuration scripts)
/etc/template.d/scripts/authentication:         'required': True}]
```

To use another "trigger" (required parameter), other than the root-password, modify e.g. `/etc/template.d/scripts/network` to make e.g. the IP address a required parameter. As mentioned

at least one required parameter is needed and if there are multiple the trigger will happen once all are received.

Once the above message gets sent, the `ovm-template-config` scripts will set up all the values and the virtual machine will end up in a configured state.

Use case: Automated Virtual Machine Provisioning

Demonstrating how virtual machines can be automatically provisioned by using Oracle VM Templates and Oracle VM Guest Additions, will be done by means of a use case where the hostname, network settings and root user password will be automatically configured when the virtual machine boots for the first time after creation.

Step by step: Creating an Oracle VM Template with Guest Additions

This chapter is a step-by-step guide explaining how to create an Oracle VM Template from scratch and how to install and configure the Oracle VM Guest Additions. The latest available Oracle Linux 7 will be used, which is OL 7.2 at the time of writing. A similar approach can be followed for Oracle Linux 5 and 6. Where there are significant differences in the installation procedure they will be mentioned.

Typically users will skip this part because ready-to-go Oracle VM Templates can be downloaded from [Oracle Software Delivery Cloud](#), e.g. Oracle Linux 7.2 Oracle VM Template (V100364-01). Like in all recent Oracle VM Templates the Oracle VM Guest Additions are already installed.

This white paper focuses on automated provisioning by using the Oracle VM Guest Additions, for best practices about Oracle VM Templates like disk structure, template specifications, how to package them,... refer to previous Oracle VM white papers on [Oracle Technology Network](#).

- **Step 1:** Create a virtual machine with a default installation of Oracle Linux 7.2.
- **Step 2:** Enable the Oracle Linux add-on channel.
Download the latest public-yum repository file from [Oracle's Public YUM repository](#) which contains more repositories and enable the add-on channel which contains the Oracle VM Guest Additions package:

```
# cd /etc/yum.repos.d
# rm public-yum-ol7.repo <-- (replace the original version with this newer version)
# wget http://public-yum.oracle.com/public-yum-ol7.repo
```

Oracle Linux updates are freely available on [Oracle's Public YUM repository](#) and the default install of Oracle Linux 7.2 already points to this location for updates.

Edit the `public-yum-ol7.repo` file to enable the `ol7_addons` channel; example:

```
[ol7_addons]
name=Oracle Linux $releasever Add ons ($basearch)
baseurl=http://yum.oracle.com/repo/OracleLinux/OL7/addons/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

Note: For other Oracle Linux releases, public-yum repo file is located at:

Oracle Linux 5: <http://public-yum.oracle.com/public-yum-el5.repo>

Oracle Linux 6: <http://public-yum.oracle.com/public-yum-ol6.repo>

If you want to install latest UEK4, available for both Oracle Linux 6 and Oracle Linux 7, you have to edit `public-yum-ol{release}.repo` and enable `ol{release}_UEK4` channel; example:

```
[ol7_UEK4]
name=Latest Unbreakable Enterprise Kernel Release 4 for Oracle Linux $releasever
($basearch)
baseurl=http://yum.oracle.com/repo/OracleLinux/OL7/UEK4/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

- **Step 3:** As a best practice update the virtual machine to the latest version of UEK and to the latest patches. Once done reboot the virtual machine.

```
# yum update
# reboot
```

- **Step 4:** Install the Oracle VM Guest Additions package. These are available for Oracle Linux 5,6 and 7. For more detailed information, see earlier on or refer to the documentation on [Oracle VM Utilities](#).

```
# yum install ovmd xenstoreprovider python-simplejson ovm-template-config
```

- **Step 5:** Install additional Oracle VM Template configuration packages:

```
ovm-template-config-authentication : Oracle VM template auth configuration script
ovm-template-config-datetime       : Oracle VM template datetime configuration script
ovm-template-config-firewall       : Oracle VM template firewall configuration script
ovm-template-config-network        : Oracle VM template network configuration script
ovm-template-config-selinux        : Oracle VM template selinux configuration script
ovm-template-config-ssh            : Oracle VM template ssh configuration script
ovm-template-config-system         : Oracle VM template system configuration script
ovm-template-config-user           : Oracle VM template user configuration script
```

For demo purposes all these additional packages can be installed, although strictly speaking not all are needed.

```
# yum install ovm-template-config-*
```

- **Step 6:** Enable and start `ovmd.service`, to be able to send and receive messages between the virtual machine and Oracle VM Manager, and `ovm-template-initial-config.service` to have template configuration scripts available for the next virtual machine boot.

```
# systemctl enable ovmd.service
# systemctl enable ovm-template-initial-config.service
# systemctl start ovmd.service
# systemctl start ovm-template-initial-config
```

Note: For other Oracle Linux releases, like 5 and 6, `systemd init system` is not available and Linux services are managed using `sysv init system`; on these Oracle Linux releases both services mentioned above are part of one unique Linux service. Enable and start it with:

```
# chkconfig ovmd on
# service ovmd start
```

After enabling `ovmd` the IP address of the virtual machine will be displayed in the Oracle VM Manager console.

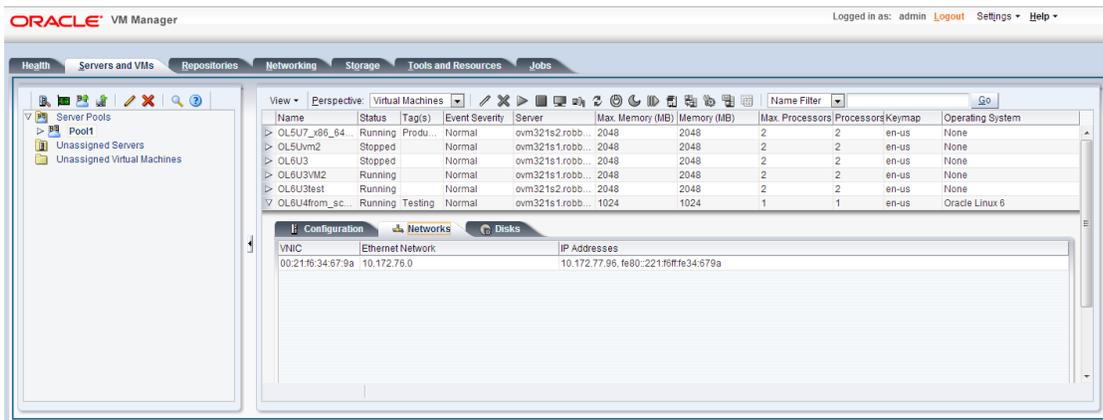


Figure 7. Guest IP address information in Oracle VM Manager UI

- **Step 7:** Now that all configuration packages are installed, scripts can be selectively enabled and disabled. This works very similar to the `chkconfig` command. To check which scripts/modules are registered and whether they are enabled to run at configure time and/or cleanup time, execute following command:

```
# ovm-chkconfig --list
name          configure unconfigure reconfigure cleanup suspend resume migrate shutdown
authentication on:90      off          off          off          off          off          off          off          off
datetime      on:50      off          off          on:50        off          off          off          off
firewall      on:41      off          off          off          off          off          off          off
network       on:50      off          off          on:50        off          off          off          off
selinux       on:30      off          off          off          off          off          off          off
ssh           on:70      off          off          on:30        off          off          off          off
system        on:60      off          off          on:60        off          off          off          off
user          on:60      off          off          on:40        off          off          off          off
#
# ovm-chkconfig --add authentication <-- (to enable all targets supported by a module)
# ovm-chkconfig --del datetime <-- (to disable all targets supported by a module)
# ovm-chkconfig --target=cleanup user off <-- (to enable or disable particular targets for a module)
```

- The two main targets are `configure` and `cleanup`. There are other targets available but they are not yet implemented at the time of writing. For the use case to configure the hostname, the network settings and the root password when the

virtual machine boots for the first time, the network module needs to be enabled when the virtual machine boots, i.e. at configure time. Make sure authentication is also enabled at configuration time to configure the root password, this to have a required parameter. To enable the authentication and network module, in case this isn't done already, execute following command and verify they are enabled:

```
# ovm-chkconfig --target configure authentication on
# ovm-chkconfig --target configure,cleanup network on
# ovm-chkconfig --list
```

- **Step 8:** For the Template configuration that is provided, and depending on optional scripts that are installed by the user, there is a well-defined set of variables (keys) that can be set. To get the list of configuration keys in a readable format execute following command:

```
# ovm-template-config --human-readable --enumerate configure
```

or, to get only a subset belonging to the network configure script:

```
# ovm-template-config --human-readable --enumerate --script network configure
```

For the use case the following keys are interesting:

```
com.oracle.linux.hostname           : System host name, e.g. "localhost.localdomain"
com.oracle.linux.network.host.0     : Hostname entry for /etc/hosts, e.g.,
                                     "127.0.0.1 localhost.localdomain localhost".
com.oracle.linux.network.device.0   : Network device to configure, e.g. "eth0"
com.oracle.linux.network.onboot.0   : Activate interface on system boot: yes or no
com.oracle.linux.network.bootproto.0 : Boot protocol: dhcp or static
com.oracle.linux.network.ipaddr.0   : IP address of the interface
com.oracle.linux.network.netmask.0  : Netmask of the interface.
com.oracle.linux.network.gateway.0  : Gateway IP address
com.oracle.linux.network.dns-servers.0 : DNS servers separated by comma, e.g.,
                                     "8.8.8.8,8.8.4.4"
```

- **Step 9:** Now that the Template configuration is done the virtual machine needs to be configured for first-boot. Run these commands on the virtual machine console because we enabled the network configuration to be cleaned, meaning network connectivity to the virtual machine will be lost.

```
# ovmd -s cleanup                <-- (reinitializes/cleans up the Template)
# sed -i 's/^INITIAL_CONFIG=.* /INITIAL_CONFIG=yes/g' /etc/sysconfig/ovm-template-initial-
config                          <-- (enables the first-boot configuration)
# shutdown -h now
```

Note: For other Oracle Linux releases, like 5 and 6, following commands need to be executed:

```
# ovmd -s cleanup                <-- (reinitializes/cleans up the Template)
# service ovmd enable-initial-config
# shutdown -h now
```

After cloning this virtual machine or starting it, it will act as a first time boot virtual machine and it will require configuration input on the virtual machine console or through the Oracle VM API. For more details about this see the following chapters.

To create multiple instances of the virtual machine prepared as Template it can be cloned to a new virtual machine or Template. For more detailed information, refer to the Oracle VM documentation on [Cloning a Virtual Machine or Template](#).

Local Configuration via Virtual Machine Console

Oracle VM Template configuration can be done locally by entering the parameter values via the virtual machine console when the virtual machine boots the first time.

- **Step 1:** After starting the virtual machine for the first time the Oracle VM Template configuration wizard will be started and the user will be able to specify values for hostname, network settings and the system root password.

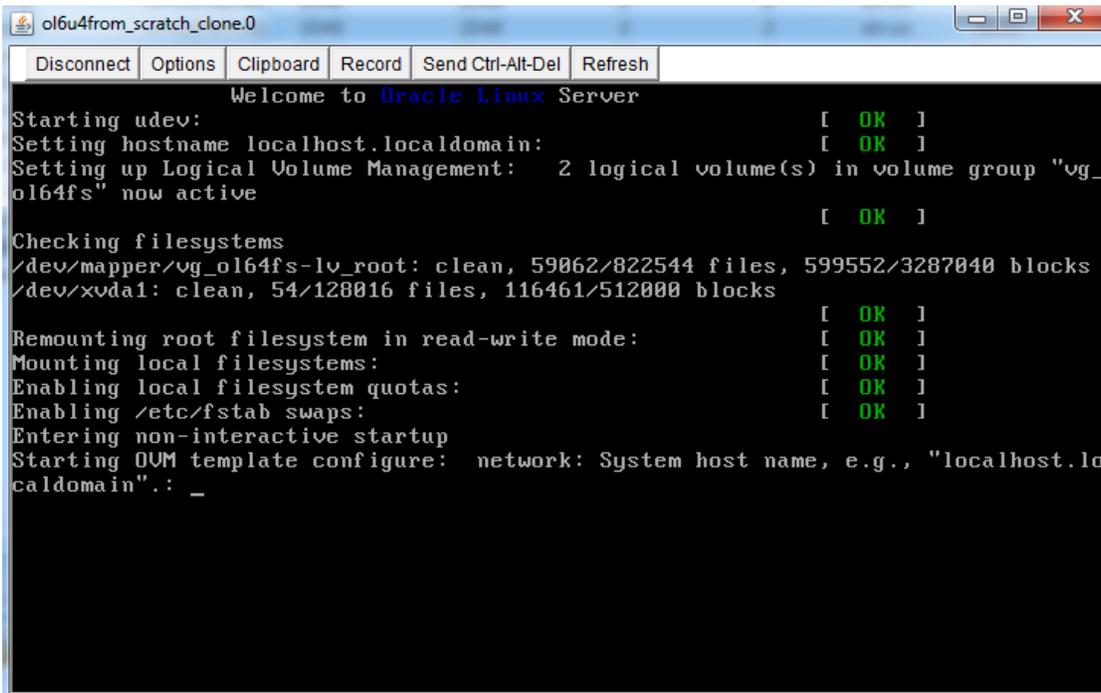
A screenshot of a virtual machine console window titled 'ol6u4from_scratch_clone.0'. The window has a menu bar with 'Disconnect', 'Options', 'Clipboard', 'Record', 'Send Ctrl-Alt-Del', and 'Refresh'. The main area is a black terminal with white text. The text shows the 'Welcome to Oracle Linux Server' message and the output of various system initialization commands, including 'Starting udev:', 'Setting hostname localhost.localdomain:', 'Setting up Logical Volume Management:', 'Checking filesystems', 'Remounting root filesystem in read-write mode:', 'Mounting local filesystems:', 'Enabling local filesystem quotas:', 'Enabling /etc/fstab swaps:', 'Entering non-interactive startup', and 'Starting OVM template configure: network: System host name, e.g., "localhost.localdomain":'. Each command is followed by a green '[OK]' status indicator. The prompt character is a dash '_'.

Figure 8. Oracle VM Template Configuration via the Virtual Machine Console

As mentioned before, it's only the root-password that is required and that needs to be filled in, all other parameters are optional and can be skipped by pressing "enter".

- **Step 2:** Once the root-password has been filled in the actual configuration will be started and the boot process completed.

```

ol6u4from_scratch_clone.0
[ Disconnect | Options | Clipboard | Record | Send Ctrl-Alt-Del | Refresh ]
Welcome to Oracle Linux Server
Starting udev: [ OK ]
Setting hostname localhost.localdomain: [ OK ]
Setting up Logical Volume Management: 2 logical volume(s) in volume group "vg_
ol64fs" now active [ OK ]
Checking filesystems
/dev/mapper/vg_ol64fs-lv_root: clean, 59062/822544 files, 599552/3287040 blocks
/dev/xvda1: clean, 54/120016 files, 116461/512000 blocks [ OK ]
Remounting root filesystem in read-write mode: [ OK ]
Mounting local filesystems: [ OK ]
Enabling local filesystem quotas: [ OK ]
Enabling /etc/fstab swaps: [ OK ]
Entering non-interactive startup
Starting OUM template configure: network: System host name, e.g., "localhost.lo
caldomain": ol64console.test.com
network: Network device to configure, e.g., "eth0": eth0
network: Activate interface on system boot: yes or no.: yes
network: Boot protocol: dhcp or static.: static
network: IP address of the interface.: 10.1
network: Netmask of the interface.: 255.255.254.0
network: Gateway IP address.: 10.1
network: DNS servers separated by comma, e.g., "8.8.8.8,8.8.4.4": 192.
authentication: System root password.: _

```

Figure 9. Oracle VM Template Configuration System Root Password

Note: The root-password is validated by the cracklib-check command meaning the password must meet several acceptance criteria. The password needs to be hard to guess:

- it needs to be minimum 6 characters long
- it cannot be a dictionary word (like "ora123")
- it needs to have enough different characters ("or12or" won't be accepted)

When the configuration wizard has been completed successfully, the next time the virtual machine boots this configuration wizard won't show up anymore.

Remote Configuration via Oracle VM CLI

Oracle VM Template configuration can also be done remotely through the Oracle VM API. In the following steps the same information (hostname, network settings, root password) will be entered by using the Oracle VM Command Line Interface (CLI) without any manual intervention via the virtual machine console. For more information on how to use the Oracle VM CLI refer to the [Oracle VM Command Line Interface User's Guide](#).

- **Step 1:** Create and boot a virtual machine that is cloned from the Template. To follow the progress of the boot process the virtual machine console can be used. The configuration wizard will be launched but unlike in the previous chapter nothing will be inputted.
- **Step 2:** Start the Oracle VM CLI on the Oracle VM Manager:

```
# ssh admin@localhost -p 10000
admin@localhost's password: <-- (admin password for the Oracle VM Manager)
OVM>
```

- **Step 3:** Send the network parameters and at the end the (required) root-password.

```
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.hostname
message=ol6u4.test.com log=no
Command: sendVmMessage vm name=ol6u4clone key=com.oracle.linux.hostname
message=ol6u4.test.com log=no
Status: Success
Time: 2013-03-15 15:36:38,574 CET
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.host.0
message="192.168.1.97 ol6u4.test.com ol6u4" log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.device.0 message=eth0
log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.onboot.0
message=yes log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.bootproto.0
message=static log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.ipaddr.0
message=192.168.1.97 log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.netmask.0
message=255.255.255.0 log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.gateway.0
message=192.168.1.1 log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.network.dns-servers.0
message=192.168.1.1 log=no
OVM> sendVmMessage vm name=ol6u4clone key=com.oracle.linux.root-password
message=password123 log=no
```

After the root-password has been send the actual configuration will be executed and the boot process continued. Once completed the virtual machine is ready to go and can be accessed through ssh.

Automated Configuration via Expect

To automate the virtual machine provisioning Expect can be used, which is a tool for automating interactive applications. The expect script can be launched from Oracle VM Manager.

- **Step 1:** After logging in to the Oracle VM Manager through ssh, install expect

```
yum install expect
```

Note: this command will also install package `tc1`

- **Step 2:** Example expect scripts are located on Oracle VM Manager under `/u01/app/oracle/ovm-manager-3/ovm_cli/expectscripts/`

- **Step 3:** To run the same CLI commands as in the previous chapter, create a demo expect script provision.exp with following content:

```
#!/usr/bin/expect
source commonExpectDef.cli
set prompt "OVM> "
set timeoutValue 3

set successMsg "Status: Success"
set failureMsg "Status: Failure"
set failFlag "False"

##### Variables to set #####
set ovmUser admin
set adminServer localhost
set ovmPassword password123
set vmName ol6u4
set hostname ol6u4.test.com
set host {"192.168.1.97 ol6u4.test.com ol6u4"}
set device eth0
set onboot yes
set bootproto static
set ipaddr 192.168.1.97
set netmask 255.255.255.0
set gateway 192.168.1.1
set dnsserver 192.168.1.1
set rootpassword password123

##### Execute CLI Commands #####
log user 1
sendSshLoginCommand $ovmUser $adminServer $ovmPassword $prompt
log file -a /tmp/ovmclilog

send user "\n## Starting Script... ##\n"

send user "## 1. Configure hostname\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.hostname message=$hostname
log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure hostname command"
$timeOutValue

send user "## 2. Configure host\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.host.0 message=$host
log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure host command"
$timeOutValue

send user "## 3. Configure device\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.device.0 message=$device
log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure device command"
$timeOutValue

send user "## 4. Configure onboot\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.onboot.0 message=$onboot
log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure onboot command"
$timeOutValue

send user "## 5. Configure boot protocol\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.bootproto.0
message=$bootproto log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure boot protocol command"
$timeOutValue

send user "## 6. Configure IP address\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.ipaddr.0 message=$ipaddr
log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure IP address command"
$timeOutValue
```

```

send user "## 7. Configure netmask\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.netmask.0
message=$netmask log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure netmask command"
$timeOutValue

send user "## 8. Configure gateway\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.gateway.0
message=$gateway log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure gateway command"
$timeOutValue

send user "## 9. Configure DNS server\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.network.dns-servers.0
message=$dnsserver log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure DNS server command"
$timeOutValue

send user "## 10. Configure root-password\n"
send "sendVmMessage vm name=$vmName key=com.oracle.linux.root-password
message=$rootpassword log=no\r"
validateCommandOutput $successMsg $failureMsg $prompt "\nConfigure root-password command"
$timeOutValue

send user "## 11. Close Session\n"
send user "\n\nScript executed successfully.\n";
send "exit\r"

```

Note: We are using `commonExpectDef.cli` to use some functions to log in, validate the output of a CLI command so you need to run your expect script from within the directory it's located
 /u01/app/oracle/ovm-manager-3/ovm_cli/expectscripts/createdelatescripts/

Conclusion

Oracle VM Guest Additions provides the tools to allow bidirectional communication directly between Oracle VM Manager and the operating system running within the guest virtual machine. This is a powerful tool that provides administrators fine-grained control over the configuration and behavior of components running within the virtual machine directly from Oracle VM Manager. Together with Oracle VM Templates, first-boot installation configuration can be performed either manually, locally from the virtual machine console or in an automated way, remotely through the messaging interface provided by the Oracle VM API.

Appendix (A): Testing of Template Configuration Scripts

Testing Template configuration scripts on a per script basis can be done in several ways. Just keep in mind at least 1 required key (e.g. the root password) is needed to trigger some action.

- *Option (1):* Configuration can be done directly from the virtual machine console by running the script with the `--console-input` option. This will prompt for values for each of the keys that need to be defined for any enabled modules:

```
# ovm-template-config -s authentication --console-input configure
```

- *Option (2):* After defining key-value pairs they can be passed to a Template configuration script with the following command:

```
# ovmd -l | ovm-template-config -s authentication --stdin configure
```

```
--stdin                build parameters from standard input
-s SCRIPT, --script=SCRIPT  specify a script name
```

Demo:

```
# ovmd -l
{"test":"something"}
# ovmd -l | ovm-template-config -s authentication --stdin configure
missing value for key "com.oracle.linux.root-password" of script "authentication"
<-- (value for the required key is missing)
#
# ovmd -p com.oracle.linux.root-password=password123          <-- (root password is set)
# ovmd -l
{"test":"something"}
{"com.oracle.linux.root-password":"password123"} <-- (root password is correctly set)
# ovmd -l | ovm-template-config -s authentication --stdin configure
<-- (runs the authentication configure script)
#
<-- (now the new root password is configured)
```

Option 3

- *Option (3):* Key-value pairs are actually passed to `ovm-template-config` in JSON format. They can be passed to a Template configuration script by creating a file e.g. `/tmp/sample.json` with the following content:

```
{
"com.oracle.linux.root-password":"password123"
}
```

When there is more than one key `sample.json` looks like:

```
{
"com.oracle.linux.hostname":"ol4u6.test.com",
"com.oracle.linux.root-password":"password123"
}
```

To run the authentication configure script with `sample.json` as input

```
# /etc/template.d/scripts/authentication configure < test.json
{"com.oracle.linux.root-password": "password123"}
```



CONNECT WITH US



[Blogs.oracle.com/virtualization](http://blogs.oracle.com/virtualization)

[Facebook.com/OracleVirtualization](https://www.facebook.com/OracleVirtualization)

[Twitter.com/ORCL_Virtualize](https://twitter.com/ORCL_Virtualize)

oracle.com

Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

Hardware and Software, Engineered to Work Together

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0316

Oracle VM 3: Oracle VM Templates Automated Virtual Machine Provisioning
March 2016
Author: Simon Cotter, Robbie De Meyer
Revision: 7.1

