HP Operations Orchestration

for the Linux operating systems

Software Version: 9.00.06

Linux KVM Integration Guide

Document Release Date: October 2011

Software Release Date: October 2011

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Web site at:

www.hp.com/go/hpsoftwaresupport

This Web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support Web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

1 Introduction

This section includes the following topics:

- Purpose of the Linux KVM Integration
- Prerequisites
- Supported Versions
- Downloading OO Releases and Documents on HP Live Network

Purpose of the Linux KVM Integration

With this integration, administrators can create HP Operations Orchestration (OO) flows that can be used to communicate with individual KVM hosts that are not managed by another product (for instance, RHEV-M).

To learn how to create OO flows, see the *Studio Guide to Authoring Operations Orchestration Flows* (Studio_AuthorsGuide.pdf) in the documentation set for the current OO release,

This document explains how this integration has been implemented, and how the integration's operations and flows communicate between OO and KVM.

The JavaDoc for the Java bindings is located here:

http://libvirt.org/sources/java/javadoc/

Libvirt is supported on Linux and Windows, but the Windows version has limited functionality and is not supported in this integration.

Prerequisites

The following are prerequisites for using the Linux KVM – OO integration:

- This integration can only be run on a Linux RAS. Currently the libvirt client library does not work on a Windows RAS.
- The libvirt library must be installed on the Linux RAS. To use this integration, you must install the libvirt-client binary package on your Linux RAS machine.

  This library is available as a package for most Linux distributions.

  Here is an example of how to install the libvirt client library on a RAS running on Linux Enterprise Linux:

  ```
  [root@roo-ras900 ~]# yum install -y libvirt-client
  ```

  You can download the latest version from the D**ownloads** section on the libvirt.org site:

  http://libvirt.org/downloads.html
- The RAS needs a   client certificate to connect using the TLS transport. To connect using the TLS transport, the RAS must have a client certificate which has been signed by the same CA as the server certificates of the KVM hosts to which it is to connect.

  Details on creating these certificates can be found on the libvirt.org site:

  http://libvirt.org/remote.html#Remote_certificates

Supported Versions

Table 1 Supported Versions

Operations Orchestration Version	Linux KVM Version
9.00.06	This integration has been tested with the following combinations of libvirt and QEMU: • libvirt 0.8.7 and QEMU 0.12.1 • libvirt 0.9.4 and QEMU 0.14.0 If a newer version of libvirt is required for a specific operation, it is mentioned in the operation's description. You can check also *http://libvirt.org/hvsupport.html*, which contains a summary of all functions with their required version for each hypervisor type.

Downloading OO Releases and Documents on HP Live Network

HP Live Network provides an **Operations Orchestration Community** page where you can find and download supported releases of OO and associated documents.

To download OO releases and documents, visit the following site:

https://www.www2.hp.com/

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

On the **HP Live Network** page, click **Operations Orchestration Community**.

The Operations Orchestration Community page contains links to announcements, discussions, downloads, documentation, help, and support.

2 Getting Started with the Linux KVM Integration

This section includes the following topics:

- Installing and Configuring the Integration
- Linux KVM – OO Architecture
- Linux KVM Use Cases

Installing and Configuring the Integration

This integration uses the Java language bindings of the Libvirt management library, The libvirt package must be installed on each KVM host machine. It is available and normally bundled with RedHat RHEL 5.5, 6.x. The simplest libvirt deployment approach is to install the RedHat bundled libvirt module. After you install the libvirt package, configure and launch libvirtd (the remote daemon) on the KVM host. To learn how to configure libvirtd, go to **http://libvirt.org/remote.html#Remote_libvirtd_configuration**.

If users want to use the TLS protocol to establish libvirt connections from the RAS to KVM hosts, they must generate TLS certificates for the libvirt servers (KVM hosts) and the client (RAS machine). See **http://libvirt.org/remote.html#Remote_certificates** to learn how to generate TLS certificates.

For more information on the configuration of libvirt secure and remote access and URI format on each type of access, see:

- **http://libvirt.org/remote.html**
- **http://libvirt.org/auth.html**
- **http://libvirt.org/uri.html**

On the Linux RAS machine, the environment variable LD_LIBRARY_PATH must include the libvirt client library path.

Linux KVM — OO Integration Architecture

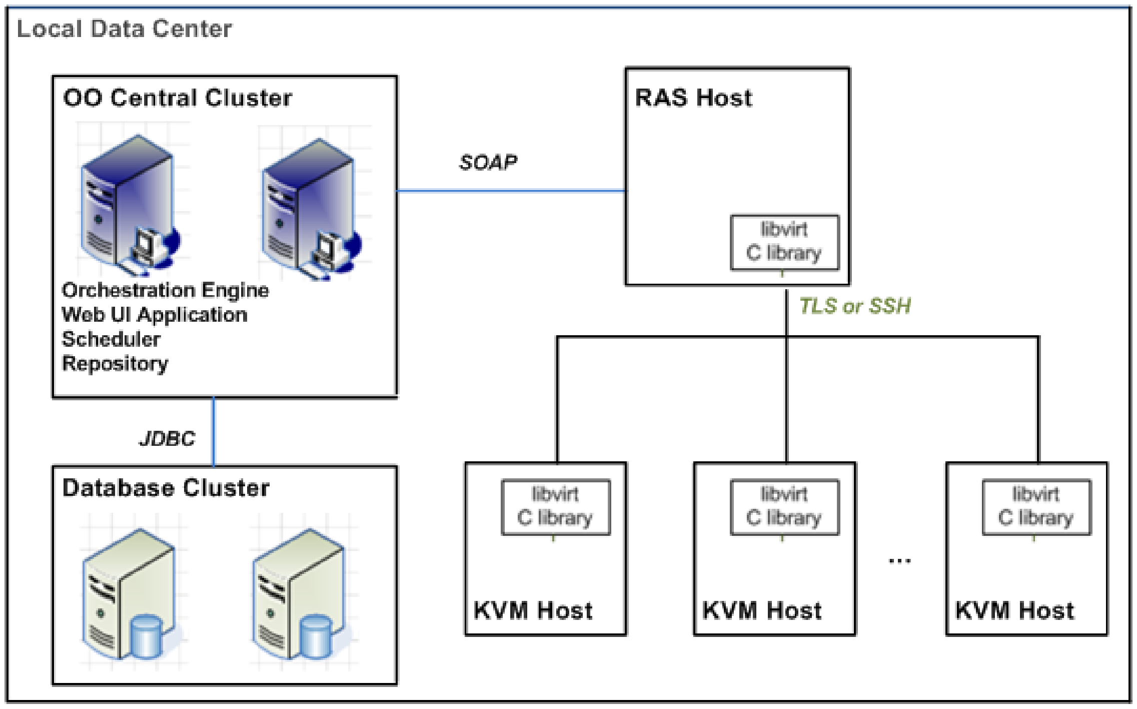


Figure 1 - Linux KVM - OO Integration Architecture

Linux KVM Use Cases

The operations and sample flows in the Library/Integrations/Linux KVM/ folder are used to manage Linux KVM virtual machines, hosts, various network and storage devices. All of these operations and flows are based on the iActions that are implemented by using the Libvirt library and its Java bindings. See **http://libvirt.org** for more information on Libvirt.

This integration can only run on a Linux RAS. Currently the libvirt client library does not work on a Windows RAS. To use this integration, the libvirt client library must be installed on the Linux RAS.

The communication between the RAS and remote KVM host is handled through libvirt, which has many options for transport protocol and authentication mechanism. At the moment there are two authentication mechanisms which work with these operations: X.509 certificate authentication used in conjunction with the TLS transport*, and public/private key authentication used with the SSH transport**.

Notes:

* The RAS needs an X.509 client certificate to connect using the TLS transport. To connect using the TLS transport, the RAS must have a client certificate which has been signed by the same CA as the server certificates of the KVM hosts to which it is to connect. Details on creating these certificates can be found on the libvirt.org site: **http://libvirt.org/remote.html#Remote_certificates**

** Due to a bug in the libvirt java bindings, password authentication with SSH and TCP protocols are unsupported.

Versions supported:

- libvirt version 0.8.7 and newer
- Older versions of Libvirt should work, but some operations will fail with a message stating that the function is not supported. There are a few operations that require a newer version of Libvirt to function correctly; this is noted in the individual operation descriptions.

Deployment Requirements:

- Linux RAS with Libvirt client libraries installed.

Following are the major use cases for the Linux KVM integration, and the operations and flows that you can use to implement them.

1 Manage KVM hosts.

  The operations in the Library/Integrations/Linux KVM/Hosts/ folder are used to obtain the general information and virtual machines list of a KVM host. The retrieved host information includes the host name, host CPUs, and memory information. This folder also includes two operations to retrieve the currently active and/or inactive VMs list of a KVM host.

  — Get Host Info

  — List Inactive VMs

  — List Running VMs

2 .Manage KVM hypervisors running on a Linux host:

  The operation in the Library/Integrations/Linux KVM/Hypervisor/ folder is used to retrieve information of the KVM hypervisor running on a host, including the type, version, capabilities, and XML description of the hypervisor.

  — Get Hypervisor Info

3 Manage interfaces.

  The operations in the Library/Integrations/Linux KVM/Interface/ folder manipulate physical host interfaces through libvirt, which uses the netcf library developed for the fedora project.

  Documentation on this set of features is scarce as of this writing, but it seems to adhere to the XML definition format as defined by the netcf project, and documentation for netcf should roughly apply to this set of libvirt operations.

  — Activate Interface

  — Deactivate Interface

  — Define Ethernet Interface

  — Define Interface From XML

  — Get Interface Info

  — List Interfaces

  — Undefine Interface

4 Manage networks.

  The Library/Integrations/Linux KVM/Networks/ folder contains operations to manipulate virtual networks on a KVM host. There are three main types of virtual network which can be created with the **Create Simple Network** operation, while more complex types of virtual network can be created using the **Create Network From XML** operation.

The simple networks can be configured to forward network traffic to the physical host interfaces using either NAT or direct routing, or can be private with no external connection. All three types can be configured with an IP address (NAT and routed require an IP), and also a DHCP server to assign IP addresses dynamically to the virtual machines connected to the virtual network.

All virtual networks can be created as either temporal or persistent, which is selected in the Create* operations using the **networkPersistent** input. The default value of the **networkPersistent** input is **true**, which creates a persistent network. Temporal networks will only exist until the next reboot of the KVM host, or until they are destroyed.

— Create Network From XML

— Create Simple Network

— Get Network Info

— Is Network Active

— List Networks

— Start Network

— Stop Network

5 Manage storage pools.

The operations in the Library/Integrations/Linux KVM/Storage Pools/ folder manipulate storage pools on a KVM host. Storage pools are used to provide the actual storage for a storage volume. A storage pool can be as simple as a folder on the KVM host, or something more complex such as an iSCSI LUN, an nfs share, or other network based storage.

— Create Storage Pool From XML

— Get Storage Pool Info

— List Storage Pools

6 Manage storage volumes.

The operations in the Library/Integrations/Linux KVM/Storage Volumes/ folder manipulate storage volumes on a KVM host. A storage volume can be as simple as a file on the KVM host, or a block device on the host which can be a storage LUN attached via SCSI, FC, iSCSI, or any other interface.

— Create Storage Volume From XML

— Delete Storage Volume

— Get Storage Volume Info

— List Storage Volumes

7 Manage virtual machines on a KVM host.

The Library/Integrations/Linux KVM/Virtual Machines/ folder contains various operations related to KVM virtual machine manipulation.

— Attach Device From XML

— Attach Disk Device

— Attach Disk Image File

- Attach Network
- Change Active VM Max MemSize
- Change Active VM Target MemSize
- Change Active VM Vcpus Count
- Configure VM Autostart
- Configure VM Parameters For Restart
- Create VM
- Create VM From XML
- Define VM
- Define VM From XML
- Detach Device by XML
- Detach Disk
- Detach Network
- Get Network Connection Info
- Get VM Info
- List Disks
- List Network Connection Info
- Migrate VM
- Reboot VM
- Resume VM
- Shutdown VM
- Start VM
- Suspend VM
- Undefine VM

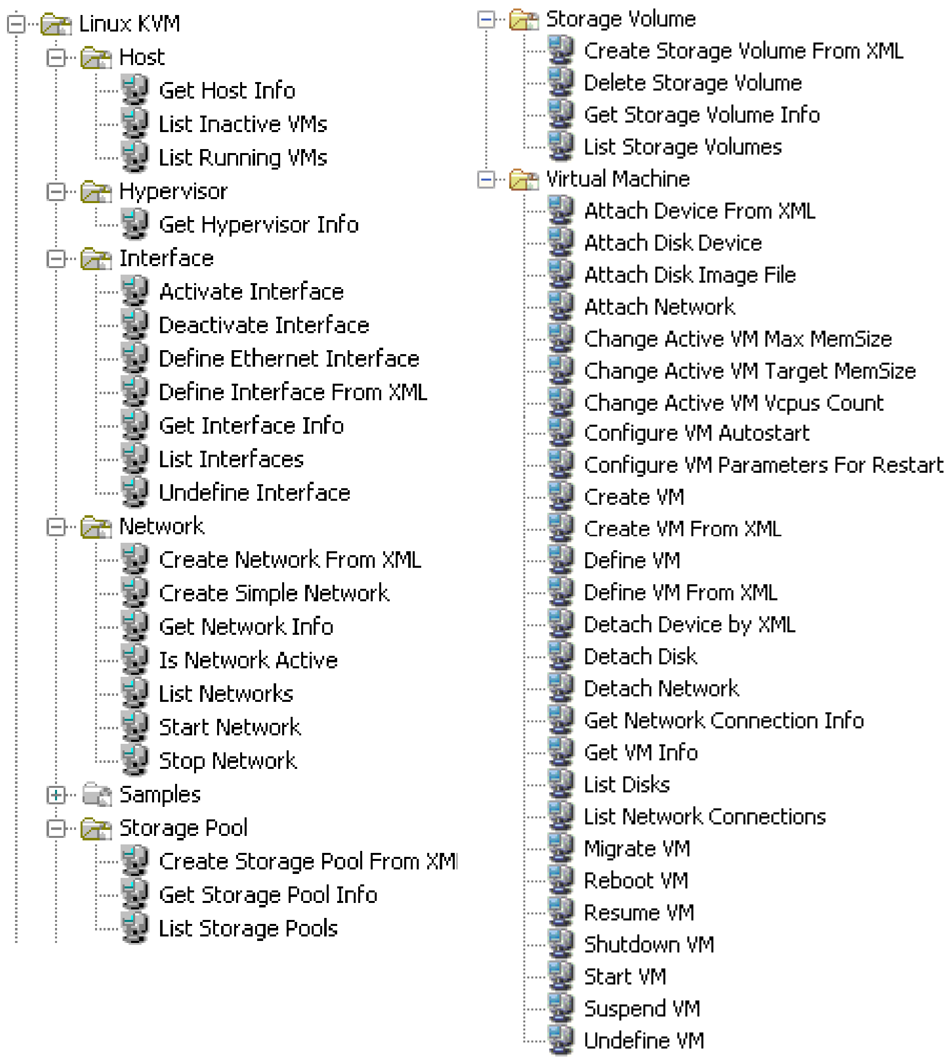
3 Using the Linux KVM – OO Integration

This section includes the following topics:

- Location of Linux KVM Integration Operations and Flows in OO Studio
- Common Inputs in the Integration
- Common Responses in the Integration
- Descriptions of Linux KVM Integration Operations and Flows

Location of Linux KVM Integration Operations and Flows in OO Studio

The Linux KVM integration includes the following operations and flows in the OO Studio Library/Integrations/Linux KVM/ folder.



- Linux KVM
  - Host
    - Get Host Info
    - List Inactive VMs
    - List Running VMs
  - Hypervisor
    - Get Hypervisor Info
  - Interface
    - Activate Interface
    - Deactivate Interface
    - Define Ethernet Interface
    - Define Interface From XML
    - Get Interface Info
    - List Interfaces
    - Undefine Interface
  - Network
    - Create Network From XML
    - Create Simple Network
    - Get Network Info
    - Is Network Active
    - List Networks
    - Start Network
    - Stop Network
  - Samples
  - Storage Pool
    - Create Storage Pool From XMl
    - Get Storage Pool Info
    - List Storage Pools
  - Storage Volume
    - Create Storage Volume From XML
    - Delete Storage Volume
    - Get Storage Volume Info
    - List Storage Volumes
  - Virtual Machine
    - Attach Device From XML
    - Attach Disk Device
    - Attach Disk Image File
    - Attach Network
    - Change Active VM Max MemSize
    - Change Active VM Target MemSize
    - Change Active VM Vcpus Count
    - Configure VM Autostart
    - Configure VM Parameters For Restart
    - Create VM
    - Create VM From XML
    - Define VM
    - Define VM From XML
    - Detach Device by XML
    - Detach Disk
    - Detach Network
    - Get Network Connection Info
    - Get VM Info
    - List Disks
    - List Network Connections
    - Migrate VM
    - Reboot VM
    - Resume VM
    - Shutdown VM
    - Start VM
    - Suspend VM
    - Undefine VM

Figure 2 - Linux KVM Integration Location in Studio

Common Inputs in the Integration

OO operations and flows use inputs to specify how they obtain the data that they need and when the data is obtained. The following inputs are used consistently throughout the Linux KVM integration's operations and flows.

uri

The URI of the RAS to connect to the KVM hypervisor. If you specify a value for the **uri** input, you do not need to specify values for any other inputs except for the **password** input if you are using SSH to connect. All other input values are ignored. Use the following format:

```
driver[+transport]://[username@][host][:port]/[path][?extraparameters]
```

Any elements that are contained within square brackets (`[]`) are optional. The default value is **qemu:///system**.

Example local **uri** values:

- `qemu:///system`
- `qemu:///session`
- `test:///default`
- `qemu+unix:///system?socket=/opt/libvirt/run/libvirt/libvirt-sock`

Example remote **uri** values:

- `qemu://<host>/system`
- `qemu+ssh://<username>@<host>[:port]/system`

driver

The hypervisor driver to which the RAS connects. The valid values are **qemu** and **test**. The default value is **qemu**.

transport

The transport that the RAS uses to connect to the KVM host. The valid values are **tls** (uses authenticated and encrypted TCP/IP socket), **tcp** (uses unencrypted TCP/IP socket), **ssh** (transports over an ordinary SSH connection), and **unix** (uses Unix domain socket, only accessible on the local machine). The default value is **tls**.

host

The KVM host name to which the RAS connects (for example, **scratchy.bar.com** and **localhost**). The default value is **localhost**.

port

The port number for the RAS to connect to the KVM host. The default value depends on which transport you use:

- **16514** for **tls**
- **16509** for **tcp**
- **22** for **ssh**

path

The connection mode that the RAS uses. The valid values are:

- For the **qemu** driver, the valid values are **system** and **session**. The default value is **system**.
- For the **test** driver, the valid value is **default**.

extraparameters

You can add extra parameters to the **uri** input. See *Extra Parameters* on the *libvirt Remote Support* page.for more information about **extraparameters** values. The extra parameters are:

- **name** - The name passed to the remote **virConnectOpen** function.
- **command** - The external command.
- **socket** - The path to the Unix domain socket, which overrides the compiled-in default.
- **netcat** - The name of the netcat command on the remote machine.
- **keyfile** - The name of the private key file to use to authentication to the remote machine.
- **no_verify** - SSH: If set to a non-zero value, this disables client's strict host key checking making it auto-accept new host keys. Existing host keys will still be validated. TLS: If set to a non-zero value, this disables client checks of the server's certificate. Note that to disable server checks of the client's certificate or IP address you must change the libvirtd configuration.
- **no-tty** - If set to a non-zero value, this stops ssh from asking for a password if it cannot log in to the remote machine automatically (for example, using SSH-agent)..
- **pkipath** - Specifies x509 certificates path for the client.

username

The username used to connect to the KVM host. Use the **username** input only when you specify a value of **ssh** for the **transport** input.

password

The password used to connect to the KVM host. Use the **password** input only when you specify a value of **ssh** for the **transport** input.

closeSession

Specifies whether to use the flow session context to cache the connection to the host. The valid values are **true** and **false**. The default value is **true**.

If you specify a value of **false** or the **closeSession** input, the steps of the flow will try to use the flow session context to cache the connection to the KVM host and to use the cached connection in the later steps of the same flow. However, for the last step in a flow you must specify a value of **true** for the **closeSession** input.

If you do not close the session in the last step of your flow you will lose the reference to the open session, and the session will remain open and in memory on the RAS. This can cause serious issues with the RAS server if sessions are not closed properly. For this reason, the default value of the **closeSession** input is **true** (session will not be cached). Use this option with caution.

Common Responses in the Integration

The following responses are used consistently throughout the Linux KVM integration's operations and flows.

success

The operation completed successfully.

failure

The operation failed.

Descriptions of Linux KVM Integration Operations and Flows

This section describes the Linux KVM integration's operations and flows, including their inputs, results, responses, examples, and other important information.

There are also a number of sample flows in the Library/Integrations/Linux KVM/ folder that demonstrate how to use certain KVM operations.

Host

Get Host Info

The **Get Host Info** operation gets the KVM host information, including its name, CPU, and memory information.

Inputs

All of the operation's inputs are described in *Common Inputs in the Integration*.

Results

The operation returns the following results:

returnResult

This is the primary output, containing general information about the KVM host.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

hostModel

The CPU model of the host.

hostMemorySize

The size of the host memory in KB.

hostCpus

The number of CPUs the host has.

hostCpuSpeed

The host CPU frequency in Mhz.

hostNodes

The number of nodes the host has.

hostSockets

The number of sockets the host has.

hostCores

The number of cores the host has.

hostThreads

The number of threads the host has.

List Inactive VMs

The **List Inactive VMs** operation lists the defined but inactive VMs in the KVM host.

Inputs

All of the operation's inputs are described in *Common Inputs in the Integration*.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the list of names of the inactive VMs.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmNames

The list of names of the inactive VMs.

vmUuids

The list of UUIDs of the inactive VMs.

List Running VMs

The **List Running VMs** operation lists the running VMs in the KVM host.

Inputs

All of the operation's inputs are described in *Common Inputs in the Integration*.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the list of names of the running VMs.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmNames

The list of names of the running VMs.

vmUuids

The list of UUIDs of the running VMs.

vmIds

The list of ID numbers of the running VMs.

Hypervisor

Get Hypervisor Info

The **Get Hypervisor Info** operation gets the KVM hypervisor information, including type, version, and the XML description of the hypervisor's capabilities.

Inputs

All of the operation's inputs are described in *Common Inputs in the Integration*.

Results

The operation returns the following results:

returnResult

This is the primary output, containing general information about the KVM host.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

type

The type of the hypervisor.

version

The version of the hypervisor.

capabilities

The XML description of the hypervisor's capabilities.

Interface

➤ Operations in this folder manipulate physical host interfaces through libvirt, which uses the netcf library developed for the fedora project. The basic host interface types are supported by libvirt version 0.8.7. However, this is a new feature of libvirt, and newer versions of the libvirt library will provide better support for these operations. The interface operations also require netcf to be installed on the KVM host. This network configuration package is standard on RedHat based systems, but may be missing on other Linux distributions. In the case that netcf is missing, these operations will fail.

Activate Interface

The **Activate Interface** operation brings up the specified network interface, placing it in the active state. The interface must be defined. If the interface is already in the active state, this operation returns success and the interface stays in the active state.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

interfaceName

The name of the network interface to start. For example**, eth0**, **br1**.

Results

The operation returns the following results:

returnResult

The name of the interface.

interfaceName

The name of the interface.

macAddress

The MAC address of the interface.

xmlDescription

The XML description for the interface.

Deactivate Interface

The **Deactivate Interface** operation removes an interface from the active pool, and deactivates it on the host. This operation leaves the interface defined so that it can be started again. If the interface is already in the inactive state, this operation succeeds and the interface is left in the inactive state.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

interfaceName

The name of the interface to deactivate. For example, **eth0**, **br1**.

Results

The operation returns the following results:

returnResult

The name of the interface that was stopped, or an error message if something went wrong.

Define Ethernet Interface

The **Define Ethernet Interface** operation defines a physical host Ethernet network interface. This operation allows you to specify a static IPv4 address or you can allow the operation to get an IP address from DHCP.

This operation can define an interface that does not have physical hardware on the host and not return an error. However, if you try and start an interface that is not present on the host, the operation will fail.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

interfaceName

The name of the interface to create. For example, **eth1**, **eth2**.

interfaceStartMode

The Start mode for the interface. This option controls whether the interface is brought up at system boot time or, in the case of hotplug, brought up when the hardware is inserted in the system. The valid values are **none**, **onboot**, or **hotplug**.

ipAddress

The IPv4 address for the interface. If you do not specify a value for this input, the IP address is obtained from DHCP. The valid value must be a valid IPv4 address. For example, **192.168.0.1**.

ipPrefix

The IPv4 network prefix. This is the number of bits from the IP address used to determine the network, plus additional bits that define the subnet. This is an alternate method of specifying the netmask. If you do not specify a value for the **ipAddress** input, you must not specify a value for this input too or the operation will fail. The valid values are integers between **8** and **32**. For example, **16**, **22**, **24**.

Results

The operation returns the following results:

returnResult

The name of the new interface.

interfaceName

The name of the new interface.

macAddress

The MAC address of the new interface.

xmlDescription

The XML description for the new interface.

Define Interface From XML

The **Define Interface From XML** operation defines a new network interface on the host from an XML definition. You can use this operation to create complex interfaces that cannot be created using the **Define Interface Ethernet** operation.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

xmlDescription

The XML string that defines the interface to be created. The format of this XML snippet is defined by the netcf utility; further documentation can be found on the internet, for example:

http://www.linux-kvm.com/content/netcf-silver-bullet-network-configuration

For example:

```
<interface type="ethernet" name="eth1">
<start mode="onboot"/>
<protocol family="ipv4">
<ip address="192.168.0.5" prefix="24"/>
<route gateway="192.168.0.1"/>
</protocol>
</interface>
```

Results

The operation returns the following results:

returnResult

The name of the new interface.

interfaceName

The name of the new interface.

macAddress

The MAC address of the new interface.

xmlDescription

The XML description for the new interface.

Get Interface Info

The **Get Interface Info** operation gets the name, MAC address, and XML definition for a host's physical network interface.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

interfaceName

The name of the interface on which to retrieve information. For example, **eth0**, **br1**.

Results

The operation returns the following results:

returnResult

The name of the interface.

interfaceName

The name of the interface.

macAddress

The MAC address of the interface.

xmlDescription

The XML description for the interface.

List Interfaces

The **List Interfaces** operation gets a list of Interfaces present on a host.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

all

Specifies whether to request all interfaces on a system or only the active interfaces. The valid values are **true** for all interfaces and **fals**e for active interfaces only. The default value is **false**.

Results

The operation returns the following results:

returnResult

A JSON encoded list of interface names. If there is an error, **returnResult** contains the error message.

interfaceNames

A JSON encoded list of interface names .

Undefine Interface

Undefine a network interface. This operation removes the configuration for the specified interface from the host system. After this, all traces of the interface are removed, and it must be defined again to be used.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

interfaceName

The name of the interface on which to retrieve information. For example, **eth0**, **br1**.

Results

The operation returns the following results:

returnResult

The name of the interface which was undefined, or an error message if something went wrong.

Network

Create Network from XML

The **Create Network From XML** operation creates a network from an XML definition. After the network is created it is left in the inactive state and must be started using the **Start Network** operation before it is used.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

xmlNetworkDefinition

A string with the XML definition of the new network. The valid value is a valid XML network definition. For example:

```
<network>
   <name>default</name>
   <bridge name="virbr0" />
   <forward mode="nat"/>
   <ip address="192.168.122.1" netmask="255.255.255.0">
   <dhcp>
      <range start="192.168.122.2" end="192.168.122.254" />
   </dhcp>
   </ip>
   <ip family="ipv6" address="2001:db8:ca2:2::1" prefix="64" />
</network>
```

networkPersistent

Specifies whether the new network is persistent or not. The valid values are **true** and **false**. The default value is **true**.

networkAutostart

Specifies whether to set the **autostart** flag on the new network. The valid values are **true** and **false**. The default is **tru**e.

Note: This flag is ignored for networks that are not persistent, since the network would not be available at next boot to autostart.

Results

The operation returns the following results:

returnResult

The name of the new network. If the operation fails, **returnResult** contains an error message

networkName

The name of the new network.

networkAutostart

A Boolean flag that represents the autostart state of the new network.

networkBridgeName

The name of the bridge interface on the host that the new network uses For example, **virbr0**.

networkUuid

The UUID of the new network.

XmlDesc

The XML description of the new network.

Create Simple Network

The **Create Simple Network** operation creates a simple network from a few parameters. The network can be forwarded to the host's physical network and can have a DHCP server running that assigns IP addresses. After the network is created it is left in the inactive state and must be started using the **Start Network** operation before it is used.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

networkName

The short name for the new network. This name is used to create files on the host system to store the network information, and should contain only alphanumeric characters. The valid value must be unique among other networks on this host. For example**, default-network**, **net1234**.

networkForwardMode

The method to use, if any, to forward network traffic from this network to the host's physical network. If you specify a value of **nat** or **route** for the **networkForwardMode** input, you must also specify a value for the **networkIPAddress** and **networkIPNetmask** inputs for the network; this address is used to route traffic to the host network. The valid values are:

- **none** - Traffic on this network is not forwarded to the physical LAN.
- **nat** - Traffic on this network is forwarded using Net Address Translation, using the host's public IP address.
- **route** - Traffic on this network is routed directly to the host, not using NAT.

networkForwardDev

An optional interface name on the host. If you specify a value for this input, traffic from the new network is restricted to the specified device. If you do not specify a value for this input, traffic from the new network can access any interface on the host. The valid value is a valid interface on the host system. For example**, eth0**, **br0**.

networkIPAddress

The IP address for this network. This is mainly used in conjunction with the **dhcpRangeStart** and **dchpRangeStop** inputs. When a DHCP server is configured, this IP address is the source for the DHCP leases. It is also used for the gateway on networks that have forwarding enabled. The valid value is a valid IP address. For example, **192.168.1.1**.

networkIPNetmask

The netmask used with the IP address specified in the **networkIPAddress** input. If you specify an IP address, you must also specify a value for the **netmaskIPNetwork** input. If you specify a value for the **networkIPNetmask** without specifying a value for the **networkIPAddress** input, the operation will fail. The valid value is a valid IP netmask. For example,**255.255.255.0**.

dhcpRangeStart

The first IP address in the range of addresses that are assigned via DHCP. Specifying a value for this input enables the DHCP server for this network, and requires that you also specify a value for the **dhcpRangeStop** input. If you specify the DHCP range, you must also specify a value for the **networkIPAddress** input. The valid value is a valid IP address. For example, **192.168.1.100**.

dhcpRangeStop

The last IP address in the range of addresses that are assigned via DHCP. The valid value is a valid IP address that is larger than the IP address you specify for the **dhcpRangeStart** input. For example, **192.168.1.200**.

networkDomain

The domain name that is assigned via the DHCP server. The valid value is any valid domain name. For example, **somenet.foo.org**.

networkPersistent

Specifies whether the new network is persistent or not. The valid values are **true** and **false**. The default is **true**.

networkAutostart

Specifies whether to set the **autostart** flag on the new network. **Note:** This flag is ignored for networks that are not persistent, since the network would not be available at next boot to autostart. The valid values are **true** and **false**. The default is **true**.

Results

The operation returns the following results:

returnResult

The name of the new network. If the operation fails, **returnResult** contains the error message

networkName

The name of the new network.

networkAutostart

A Boolean flag representing the autostart state of the new network.

networkBridgeName

The name of the bridge interface on the host that the new network uses. For example, **virbr0**.

networkUuid

The UUID of the new network.

XmlDesc

The XML description of the new network.

Get Network Info

The **Get Network Info** operation gets information for a virtual network.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

networkName

The name of the network for which to retrieve information. The valid value is any defined or active network on the KVM host. For example, **default**, **some-network**.

Results

The operation returns the following results:

returnResult

The name of the network. If the operation fails, **returnResult** contains the error message

networkName

The name of the network.

networkAutostart

A Boolean flag determining the autostart state of the interface.

networkBridgeName

The name of the bridge interface on the host that this interface uses. For example, **virbr0**.

networkUuid

The UUID of the network

networkXmlDesc

The XML description of the network.

Is Network Active

The **Is Network Active** operation gets the status for a virtual network.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

networkName

The name of the network for which to retrieve information. The valid value is any defined or active network on the KVM host. For example, **default**, **some-network**.

Results

The operation returns the following results:

returnResult

Contains **true** if the network is active or f**alse** if it is not. **returnResult** contains an error message if something goes wrong.

isActive

Contains true if the network is active or f**alse** if it is not.

List Networks

The **List Networks** operation gets information for a virtual network.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

all

Specify a value of **true** to list all networks or **false** to list just the active ones. The default is **false**.

Results

The operation returns the following results:

returnResult

A JSON encoded list of network names.

networkNames

A JSON encoded list of network names

Start Network

The **Start Network** operation sets a virtual networks state to active.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

networkName

The name of the network to start. This network must be inactive, or the operation returns an error. The valid value is any defined or inactive network on the KVM host. For example, **some-network**.

Results

The operation returns the following results:

returnResult

This result is empty if the network is started, or contains an error message if something goes wrong.

Stop Network

The **Stop Network** operation sets a virtual networks state to inactive.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

networkName

The name of the network to stop. The network must be active or the operation returns an error. The valid value is any defined or active network on the KVM host. For example, **default**, **some-network**.

Results

The operation returns the following results:

returnResult

This result is empty if the network is stopped, or contains an error message if something goes wrong.

Storage Pool

Create Storage Pool From XML

The **Create Storage Pool From XML** operation creates a storage pool from an XML definition. For a full description of the format, and valid values, of the XML definition, see the following URL:

http://libvirt.org/formatstorage.html#StoragePool

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

xmlPoolDefinition

The XML string containing the definition for the new storage pool. For example, **<pool type='dir'> <name>default</name> <source> </source> <target> <path>/var/lib/libvirt/images</path> </target> </pool>**.

poolPersistent

Specifies whether the pool should be persistent. If you specify a value of **true**, the pool is persistent and exists after a reboot. If you specify a value of **false**, the pool disappears when it is deactivated. The valid values are **true** and **false**. The default value is **true**.

poolAutostart

Specifies whether the storage pool is started at boot time. The valid values are **true** and **false**. The default value is **true**.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it contains the error message.

poolInfo

A JSON-encoded object, containing the total capacity in bytes, current allocation in bytes, available bytes, and current state for the storage pool

poolName

The name of the new storage pool.

poolUuid

The UUID of the new storage pool.

poolAutostart

The value of the **autostart** flag for the new volume.

poolPersistent

Specifies whether the pool is persistent (will still exist after a reboot of the host).

poolState

The current state of the storage pool (**running** or **inactive**).

Example

The following is an example XML pool definition:

```
<pool type='dir'>
   <name>new-pool</name>
   <source>
   </source>
   <target>
      <path>/var/lib/libvirt/new-pool</path>
   </target>
</pool>
```

This creates a new pool named **new-pool** under the directory /var/lib/libvirt/new-pool.

For more detailed examples see the libvirt storage XML specification at **http://libvirt.org/formatstorage.html**.

Get Storage Pool Info

The **Get Storage Pool** operation gets information for a given storage pool.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

storagePoolName

The name of the storage pool for which to retrieve information. The valid value is a defined storage pool on the KVM host. For example, **default**, **some-pool**.

Results

The operation returns the following results:

returnResult

The result contains the name of the storage pool or the error message if the operation fails.

poolName

The name of the new storage pool.

poolUuid

The UUID of the new storage pool.

poolInfo

A JSON encoded object containing the total capacity in bytes, current allocation in bytes, available bytes, and current state for the storage pool.

poolAutostart

The value of the autostart flag for the new volume.

poolPersistent

A flag indicating if the pool is persistent (will still exist after a reboot of the host).

poolState

The current state of the storage pool (**running** or **inactive**).

poolXMLDesc

The XML definition of the storage pool.

List Storage Pools

The **List Storage Pools** operation lists all active, or active and inactive storage pools on the host.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

all

Specifies whether to return all pools. If you specify a value of **true**, the operation returns all active and inactive storage pools. If you specify a value of **false**, the operation returns only active storage pools. The valid values are **true** and f**alse**. The default value is **false**.

Results

The operation returns the following results:

returnResult

The result contains the name of the storage pools or the error message if the operation fails.

storagePools

A JSON-encoded list containing the name of each storage pool on the current host.

Storage Volume

Create Storage Volume From XML

The **Create Storage Volume From XML** operation creates a storage volume from an XML definition. For a full description of the format and valid values of the XML definition, go to:

http://libvirt.org/formatstorage.html#StorageVol

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

xmlVolumeDefinition

An XML string containing the complete definition for the storage volume to create.

storagePoolName

The name of the storage pool in which to create the new volume.

Results

The operation returns the following results:

returnResult

This result returns the volume name unless the operation fails, in which case it returns an error message.

volumeName

The name of the newly created volume.

volumeKey

The newly created volume's key. This is a unique identifier for the new volume.

volumeInfo

A JSON-encoded list of three parameters for the new volume:

- **capacity** - The logical capacity in bytes for the volume.
- **allocation** - The number of bytes actually allocated for the volume.
- **type** - The underlying source for the volume. The valid values are **file**, **block**, **dir**, and **network**.

volumeXmlDesc

The XML description of storage volume.

Delete Storage Volume

The **Delete Storage Volume** operation deletes a storage volume for a storage pool and removes the file from the system.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

identifierType

The type of identifier contained in the **volumeIdentifier** input. The valid values are the storage volume's path or key. The default value is the storage volume's path.

volumeIdentifier

The identifier for the storage volume. The identifier type is specified using the **identifierType** input.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it returns the error message

Get Storage Volume Info

The **Get Storage Volume Info** operation retrieves information about a storage volume, including the name and key of the volume, and the capacity and current allocation for the volume.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

identifierType

The type of identifier contained in the **volumeIdentifier** input. The valid values are the storage volume’s path or key. The default value is the storage volume’s path.

volumeIdentifier

The identifier for the storage volume. The identifier type is specified using the **identifierType** input.

Results

The operation returns the following results:

returnResult

The name of the volume.

volumeName

The name of the volume.

volumeKey

The volume's key. This is a unique identifier for the volume.

volumeInfo

A JSON-encoded list of three parameters for the volume:

- **capacity** - The logical capacity in bytes for the volume.
- **allocation** - The number of bytes actually allocated for the volume.
- **type** - An attribute that specifies the underlying source for the volume. The valid values are **file**, **block**, **dir**, and **network**.

volumeXmlDesc

The XML description of storage volume.

List Storage Volumes

The **List Storage Volumes** operation returns a list of paths, one for each storage volume contained in a given storage pool.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

storagePoolName

The name of the pool for which to get the list of volumes.

Results

The operation returns the following results:

returnResult

This result returns the value of the **volumePaths** result unless the operation fails, in which case it returns an error message.

volumePaths

A JSON-encoded list of volume paths, one for each volume in the pool.

Virtual Machine

Attach Device From XML

The **Attach Device From XML** operation attaches a virtual device to a virtual machine. The device can be any type supported by the host, and is defined by an XML string. This operation is meant to be used for more complex devices than those supported by the basic operations.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

deviceXml

The XML string that defines the new device to add to the virtual machine.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it contains the error message

Attach Disk Device

The **Attach Disk Device** operation attaches a block IO device to a virtual machine. Block devices are generally physical devices attached to a VM host, including disk drives, FC LUNs, and iSCSI LUNs.

Note: The virtual machine to which the virtual disk is to be added must be running. If you want to add a virtual disk to a VM that is not running, you must get the VM's XML description, add in the new virtual disk's XML definition to the **<devices>** section, and then redefine the VM. For more complex configurations see the **Attach Device From XML** operation for details on adding a device using an XML definition. Details on the XML definition of a virtual disk can be found in the libvirt documentation: **http://libvirt.org/formatdomain.html#elementsDisks**

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value to use to look up the virtual machine located on the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type used to look up the virtual machine.

sourcePath

The full path to the block device to be attached to the VM (for example, **/dev/sda**).

targetBus

The bus on the virtual machine to which to attach the new disk. If you do not specify a value for this input, the bus is inferred from the value of the **targetDevice** input.

targetDevice

The path to the new disk on the virtual machine. This is the device path at the hardware level, and is not guaranteed to map to the same device in the operating system running on the VM (for example, **sda**, **hda**).

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it returns an error message.

Attach Disk Image File

The **Attach Disk Image File** operation attaches an image file based disk to a virtual machine.

Note: The virtual machine to which the virtual disk is to be added must be running. If you want to add a virtual disk to a VM that is not running, you must get the VM's XML description, add in the new virtual disk's XML definition to the **<devices>** section, and then redefine the VM. For more complex configurations see the **Attach Device From XML** operation for details on adding a device using an XML definition. Details on the XML definition of a virtual disk can be found in the libvirt documentation: **http://libvirt.org/formatdomain.html#elementsDisks**

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value used to look up the virtual machine located on the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type to use to look up the virtual machine.

sourcePath

The full path to the file to be attached to the VM (for example, **/var/lib/libvirt/images/hd.img**).

targetBus

The bus on the virtual machine to which to attach the new disk. If you do not specify a value for this input, the bus is inferred from the value of the **targetDevice** input.

targetDevice

The path to the new disk on the virtual machine. This is the device path at the hardware level, and is not guaranteed to map to the same device in the operating system running on the VM (for example, **sda**, **had**).

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it returns an error message

Attach Network

The **Attach Network** operation attaches a virtual machine to a network on the host. The network to attach the virtual machine to can be a virtual network or a physical bridge interface on the host. This creates a new virtual network interface card on the VM, using the default virtual interface driver for the hypervisor running on the host.

Note: The virtual machine to which the virtual network adapter is to be added must be running. If you want to add a network connection to a VM which is not running, you must get the VM's XML description, add in the new virtual adapter's XML definition to the <devices> section, and then redefine the VM. For more complex configurations, such as when you need control of the virtual NIC type or need to create a connection to a network that is not a virtual network or bridge device on the host, see the **Attach Device From XML**

operation for details on adding a device using an XML definition. For more details on the XML definition for a virtual network connection, see the libvirt documentation **http://libvirt.org/formatdomain.html#elementsNICS**.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

networkType

The type of network to connect to the VM. The valid values are **network** and **bridge**.

networkName

The name of the virtual network or host bridge to which to connect the VM. For example, **default**, **some-network**, **br0**.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it contains the error message.

Change Active VM Max MemSize

The **Change Active VM Max MemSize** operation dynamically changes the maximum amount of physical memory (in KB) allocated to the virtual machine. This operation requires privileged access to the hypervisor. Note that this operation can only be applied to an active VM.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

vmMaxMemSize

The maximum amount of physical memory (in KB) allocated to the VM. The valid values are **524288** and **1048576**.

Results

The operation returns the following results:

returnResult

A confirmation message that the VM's maximum amount of physical memory is changed.

vmMaxMem

The maximum amount of physical memory (in KB) allocated to the VM after the change.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The current state of the VM.

connectionUri

The connection URI that RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Change Active VM Target MemSize

The **Change Active VM Target MemSize** operation dynamically changes the target amount of physical memory (in KB) allocated to the virtual machine. This operation requires privileged access to the hypervisor. You can only use this operation on an active VM.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

vmMemSize

The target amount of physical memory (in KB) allocated to the VM. The valid values are **524288** and **1048576**.

Results

The operation returns the following results:

returnResult

A confirmation message that the VM's target amount of physical memory is changed.

vmUsedMem

The VM's used memory size (in KB) after its target memory size is changed.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The current state of the VM.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Change Active VM Vcpus Count

The **Change Active VM Vcpus Count** operation dynamically changes the number of virtual CPUs used by this virtual machine. This operation will fail if the underlying KVM hypervisor does not support it or if growing the number is arbitrarily limited. This operation requires privileged access to the hypervisor, and can only be applied to an active VM.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

vmCpuCount

The number of virtual CPUs used by the VM. The valid values are **1**, **2**, and **3**.

Results

The operation returns the following results:

returnResult

A confirmation message that the VM's number of virtual CPUs used is changed.

vmVirtualCpus

The number of virtual CPUs used by the VM after change.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The current state of the VM.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Due to a RedHat bug, when this operation is used for a VM running on some RedHat Linux KVM hosts it may fail with the error message: "org.libvirt.LibvirtException: internal error unable to execute QEMU command 'cpu_set': The command cpu_set has not been found".

Configure VM Autostart

The **Configure VM Autostart** operation configures the virtual machine as to whether it should be automatically started when the host machine reboots. You can change the VM's autostart configuration no matter what the VM's state is. The operation can be applied to either an active or inactive persistent (defined) VM. Note that the operation cannot be applied to a non-persistent (transient) VM.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

vmAutostart

Specifies whether to enable or disable the automatic starting of the VM automatically when the host machine boots. The valid values are **true** and **false**.

Results

The operation returns the following results:

returnResult

A confirmation message that the VM's autostart is enabled or disabled.

vmAutostart

The VM's autostart state after the change.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The current state of the VM.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Configure VM Parameters For Restart

The **Configure VM Parameters For Restart** operation changes the VM's parameters including the virtual CPU count and maximum memory size. The changed parameters configuration does not affect the current VM instance if it is active. The new configuration will apply when the VM is shutdown and restarts. This operation can be used on either active or inactive VMs. It defines the VM with the new configuration so that the operation can also be used to make an active non-persistent VM persistent, with or without changing its parameters configuration.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

vmCpuCount

The maximum number of virtual CPUs allocated to the VM OS. The value must be between **1** and the maximum supported by the hypervisor. For example, **1**, **2**, or **3**. The default is **1**.

vmMaxMemSize

The maximum allocation of memory for the VM at boot time, in kilobytes. For example, **524288** or **1048576**. The default is **524288**.

vmDescription

Specifies a human readable description of the virtual machine. This is optional.

Results

The operation returns the following results:

returnResult

A confirmation message that the VM's parameter configuration is changed and preserved.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The current state of the VM.

vmXmlDesc

The VM's current XML description. If the VM is currently active, the value of **vmXmlDesc** does not include the changed parameters configuration that is included in **vmXmlDescNew** and the new configuration only applies when the current VM instance is shutdown and a new VM instance starts.

vmXmlDescNew

The VM's newly defined XML description with the changed parameter configuration.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Create VM

The **Create VM** operation creates and starts a non-persistent virtual machine on the KVM host, based on the VM definition parameters you provide.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmName

A short name for the virtual machine. The name should only consist of alpha-numeric characters, and must be unique within the scope of a single host.

vmDescription

A human readable description of the virtual machine. This data is not used by libvirt, so it can contain any information you want.

vmCpuCount

The maximum number of virtual CPUs allocated for the guest operating system. The value of this input must be between 1 and the maximum supported by the hypervisor (for example, **1**, **2**, or **3**). The default value is **1**.

vmMemSize

The maximum allocation of memory in kilobytes for the guest at boot time (for example, **524288** or **1048576**). The default value is **524288**.

vmOsArch

The CPU architecture to virtualization for the VM guest operating system (for example, **i686** or **x86_64**). The default value is provided by the hypervisor. The valid architecture values are available from the hypervisor capabilities description which you can obtain using the **Get Hypervisor Info** operation.

vmOsMachine

The machine type of the VM guest operating system (for example, **pc**, **rhel6.0.0** or **rhel5.5.0**). The default value is provided by the hypervisor. The valid machine values are available from the hypervisor capabilities description which you can obtain using the **Get Hypervisor Info** operation.

vmOsBootDevices

A priority list of boot devices to try in turn when starting the VM guest operating system. The valid values are **hd**, **fd**, **cdrom**, and **network**. The default is **hd**.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is created.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmName

The name of the created VM.

vmUuid

The UUID of the created VM.

vmId

The ID number of the created VM.

Known issue caused by RedHat Linux KVM Bug 708887 - qemu-kvm exits with "Too many boot devices for PC". This issue doesn't happen if you boot the guest with less than four boot devices (see **https://bugzilla.redhat.com/show_bug.cgi?id=708887** for details). To avoid the issue caused by this KVM bug, you cannot choose more than three devices for **vmOsBootDevices** input.

Create VM From XML

The **Create VM From XML** operation creates and starts a non-persistent virtual machine on the KVM host, based on the VM definition in the input XML file.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmXmlDesc

A string of the complete XML description of the VM definition. See the libvirt webpage on the requirements of the VM definition XML file format (**http://libvirt.org/formatdomain.html**).

Examples:

```
<domain type='kvm' id='3'>
  <name>myTestVM</name>
  <memory>524288</memory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.1.0'>hvm</type>
  <boot dev='hd'/>
 </os>
 <clock offset='utc'/>
 <on_poweroff>destroy</on_poweroff>
 <on_reboot>restart</on_reboot>
 <on_crash>destroy</on_crash>
 <devices>
  .........
 </devices>
 <seclabel type='dynamic' model='selinux'>
  <label>system_u:system_r:svirt_t:s0:c912,c935</label>
  <imagelabel>system_u:object_r:svirt_image_t:s0:c912,c935</imagelabel>
 </seclabel>
</domain>.
```

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is created.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmName

The name of the created VM.

vmUuid

The UUID of the created VM.

vmId

The ID number of the created VM.

Define VM

The **Define VM** operation defines (but does not start) a persistent virtual machine on the KVM host, based on the VM definition parameters you provide.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmName

A short name for the virtual machine. The name should only consist of alpha-numeric characters and must be unique within the scope of a single host. This is a required input.

vmDescription

A human readable description of the virtual machine. This data is not used by libvirt, so it can contain any information you want.

vmCpuCount

The maximum number of virtual CPUs allocated for the guest operating system (for example, **1**, **2**, or **3**). The default value is **1.** The value must be between 1 and the maximum supported by the hypervisor. This is a required input.

vmMemSize

The maximum allocation of memory in kilobytes for the guest at boot time (for example, **524288** or **1048576**). The default value is **524288.** This is a required input.

vmOsArch

The CPU architecture to virtualization for the VM guest operating system (for example, **686** or **x86_64**). The hypervisor provides a default value. The valid architecture values are available from the hypervisor capabilities description which can be obtained by the **Get Hypervisor Info** operation.

vmOsMachine

The machine type of the VM guest operating system (for example, **pc**, **rhel6.0.0**, or **rhel5.5.0**). The hypervisor provides a default value. The valid machine values are available from the hypervisor capabilities description which can be obtained by using the **Get Hypervisor Info** operation..

vmOsBootDevices

A priority list of boot devices to try in turn when starting the VM guest operating system. The valid values are **hd**, **fd**, **cdrom**, and **network**. The default is **hd**.

vmAutostart

Specifies whether to enable the VM to be automatically started when the host machine reboots. The valid values are **true** and **false**. The default value is **false**.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is created.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmName

The name of the created VM.

vmUuid

The UUID of the created VM.

Known issue caused by RedHat Linux KVM Bug 708887 - qemu-kvm exits with "Too many boot devices for PC". This issue doesn't happen if you boot the guest with less than four boot devices (see **https://bugzilla.redhat.com/show_bug.cgi?id=708887** for details). To avoid the issue caused by this KVM bug, you cannot choose more than three devices for **vmOsBootDevices** input.

Define VM From XML

The **Define VM From XML** operation defines (but does not start) a persistent virtual machine on the KVM host, based on the VM definition in the input XML file.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmAutostart

Specifies whether to enable the VM to be automatically started when the host machine reboots. The valid values are **true** and **false**. The default value is **false**.

vmXmlDesc

A string of the complete XML description of VM definition. See the libvirt webpage on the requirements of the VM definition XML file format (**http://libvirt.org/formatdomain.html**).

Examples:

```
<domain type='kvm' id='3'>
  <name>myTestVM</name>
  <memory>524288</memory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.1.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
   .........
  </devices>
  <seclabel type='dynamic' model='selinux'>
```

```
    <label>system_u:system_r:svirt_t:s0:c912,c935</label>
    <imagelabel>system_u:object_r:svirt_image_t:s0:c912,c935</imagelabel>
  </seclabel>
</domain>.
```

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is defined.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmName

The name of the defined VM.

vmUuid

The UUID of the defined VM.

Detach Device by XML

The **Detach Device by XML** operation detaches a device from a virtual machine, identified by the device's XML definition.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

deviceXml

The XML string that defines the device to remove from the virtual machine. This XML string must contain enough information to uniquely identify the device to remove. Usually you will need the full XML description of the device to remove the XML definition of the virtual machine. This can be parsed from the output of the **Get VM Info** operation.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case it contains the error message

Detach Disk

The **Detach Disk** operation detaches the specified storage volume from a virtual machine. This does not affect the file on disk.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value to use to look up the virtual machine located on the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type used to look up the virtual machine.

storageVolumePath

The source path of the storage volume to remove from the domain..

Results

The operation returns the following results:

returnResult

This returns an empty field unless the operation fails, in which case it returns an error message.

Detach Network

The **Detach Network** operation detaches a virtual network connection from a running virtual machine. The virtual network connection is identified by its target device on the virtual machine, which can be found using the **Get Network Connection Info** operation.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

networkTargetDev

The target device of the virtual NIC to detach. For example, **vnet0**, **vnet7**.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case this contains is the error message

Get Network Connection Info

The **Get Network Connection Info** operation gets detailed information on an interface that connects a VM to a virtual or physical network.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

networkTargetDev

The target device of the network to look up. This is the device name, on the VM or guest side. For example, **vnet0**, **vnet7**.

Results

The operation returns the following results:

returnResult

This result is blank unless the operation fails, in which case this contains the error message

networkType

The type of network to which this connection maps the VM.

xmlDescription

The XML definition for the interface.

macAddress

The MAC address for the virtual NIC. This is optional.

sourceName

The optional: name of the source. The meaning of this depends on the value of **networkType**. For example, for a **networkType** of **network**, the **sourceName** would be a virtual network on the host.

targetDevice

The optional target device. This is the network device on the VM or guest side for the virtual NIC.

networkAlias

The optional alias name for the **targetDevice**. **Note:** Results that are marked as optional are not guaranteed to exist for every network type. If the information does not exist for the given **networkType**, the value is an empty string.

Get VM Info

The **Get VM Info** operation gets the virtual machine information, including its identification, state, CPU, memory information, and XML description.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of value to look up on the virtual machine located on the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type used to look up the virtual machine.

Results

The operation returns the following results.

returnResult

This is the primary output, containing the general information for the VM.

vmName

The name of the VM.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The current state of the VM.

vmCpuTime

The consumed CPU time in seconds of the VM.

vmVirtualCpus

The number of CPUs allocated to the VM.

vmMaxMem

The maximum memory size in KB allocated to the VM.

vmUsedMem

The currently used memory size in KB of the VM.

vmOsType

The operating system type of the VM.

vmXmlDesc

The XML description of the VM..

vmActive

Specifies if the VM is active.

vmAutostart

Specifies whether the VM should automatically start when the host machine reboots.

vmPersistent

Specifies whether the VM's configuration should be persistent in the host.

List Disks

The **List Disks** operation lists all of the disks attached to the specified virtual machine.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value to use to look up the virtual machine located on the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

A JSON-encoded list of storage volume paths unless the operation fails, in which case this is the error message.

volumePaths

A JSON-encoded list containing the path of each volume attached to this VM.

volumeStatistics

A JSON-encoded map, with the following values for each storage volume:

- **Volume type** – This value should be VIR_STORAGE_VOL_FILE.
- **capacity** - The logical capacity in bytes for the volume.
- **allocation** - The number of bytes actually allocated for the volume.

The capacity statistics **allocation** and **capacity** will only differ if the volume is set up for sparse allocation (thin provisioning in VMware terms).

List Network Connections

The **List Connections** operation lists all network connections present on a virtual machine. You can use the list is of target devices returned by this operation with the **Get Network Connection Info** operation to retrieve more detailed information about the connection.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

A JSON encoded list of target devices, one for each network connection. If the operation fails, this result contains the error message

networkList

A JSON encoded list of target devices, one for each network connection.

Migrate VM

The **Migrate VM** operation migrates an active virtual machine from its current host to the destination host. You can choose either live migration or suspending the VM after migrating it to the destination host. You may also choose whether to change the VM's name after migration.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

newVmName

The new VM name after migration. This is optional.

vmMigrateFlags

The flags used for KVM virtual machine migration. You can choose multiple flags at the same time. The valid values are:

- **live** – Live migration.
- **paused** - The VM is paused after migration.
- **persist_dest** - Persist the migrated VM on the destination host.
- **undefine_src** - Undefine the original VM on the source host. If this option is not chosen, the source VM is shut off but keeps defined if it is originally defined at the source host.
- **peer2peer** - Use direct source->dest host control channel for migration.
- **tunneled** - Tunnel migration data over libvirtd connection.

The default value is **live**.

destUri

The URI for the RAS to connect to the destination KVM host. If you specify a value for **destUri**, you do not need to provide other destination connection inputs; other destination connection inputs are ignored except the value of the **destPassword** input if you are using **SSH** for the destination connection.

destDriver

The hypervisor driver that the RAS uses to connect to the destination host.

destTransport

The transport that the RAS uses to connect to the destination host.

destHost

The destination host name.

destPort

The port number for the destination connection.

destPath

The destination connection mode that the RAS uses.

destExtraparameters

Extra parameters for the destination connection.

destUsername

The username used to connect to the destination host.

destPassword

The password used to connect to the destination host.

Results

The operation returns the following results:

returnResult

The confirmation message that the VM is migrated.

vmNameOld

The VM name before the migration.

vmName

The VM name after the migration.

vmUuid

The UUID of the VM.

vmState

The VM state after the migration.

connectionUri

The connection URI that the RAS used to connect to the source host.

hostname

The DNS name of the source host.

destConnectionUri

The connection URI that the RAS used to connect to the destination host.

destHost

The DNS name of the destination host.

There are many limitations on VM migration imposed by the hypervisor. For example, a VM can only migrate to a destination host whose CPUs are compatible with those of the source host where the VM is currently located. Normally, the source host CPUs and the destination host CPUs for VM migration are from the same vendor, and the features of the destination host CPUs are identical to or are a superset of those of the source host CPUs

Reboot VM

The **Reboot VM** operation reboots the virtual machine. The VM's operating system is being stopped for a restart. Note that the guest operating system may ignore the request, and the operation can only work for the VMs running on the KVM hosts where libvirt version 0.9.4 is installed.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

The confirmation message that the VM's operating system is being restarted.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmState

The VM's current state.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

- The guest operating system may ignore the request from this operation.
- The operation can only work for the VMs running on the KVM hosts where libvirt version 0.9.3 or later are installed.

Resume VM

The **Resume VM** operation resumes a suspended virtual machine. The VM process is restarted from the state where it was suspended. This operation requires privileged access to the hypervisor.

Inputs

All of the operation’s inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

The confirmation message that the VM is resumed.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmStateOld

The VM state before it is resumed.

vmStateNew

The VM state after it is resumed.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Shutdown VM

The **Shutdown VM** operation shuts down a running virtual machine if it is not already down, and gives back all resources used by the VM to the hypervisor. The VM is identified by its name, UUID, or ID number.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value to use to look up the virtual machine located on the KVM host. The valid values are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type to use to look up the virtual machine.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is shut down.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Start VM

The **Start VM** operation starts the specified virtual machine on the KVM host if it is shut down. The VM is identified by its name, UUID, or ID number.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value to use to look up the virtual machine located on the KVM host. The valid values are **vmName, vmUuid**, and **vmId**.

vmInputValue

The value corresponding to the given type used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is started.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

vmName

The name of the started VM.

vmUuid

The UUID of the started VM.

vmId

The ID number of the started VM.

Suspend VM

The **Suspend VM** operation suspends the virtual machine if it is active. The VM is frozen without further access to CPU resources and I/O, but the memory used by the VM at the hypervisor level stays allocated. Use the **Resume VM** operation to reactivate the VM. This operation requires privileged access to the hypervisor. Note that this operation can only be applied to an active VM; it fails when applied to a shutoff VM.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value provided to look up the virtual machine located in the KVM host. The valid value are **vmName**, **vmUuid**, and **vmId**.

vmInputValue

The value (corresponding to the given type) used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

The confirmation message that the VM is suspended.

vmName

The name of the VM.

vmUuid

The UUID of the VM.

vmId

The ID number of the VM.

vmStateOld

The VM state before it is suspended.

vmStateNew

The VM state after it is suspended.

connectionUri

The connection URI that RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

Undefine VM

The **Undefine VM** operation undefines a persistent VM on the KVM host. The VM is identified by either its name or UUID. Only shutoff VMs can be undefined. If you want to undefine a VM with state other than **shutoff**, you must first shut it down before you undefine it. This operation requires privileged access to the hypervisor.

Inputs

All of the operation's inputs except the following are described in *Common Inputs in the Integration*.

vmInputType

The type of the value to use to look up the virtual machine located on the KVM host. The valid values are **vmName** and **vmUuid**.

vmInputValue

The value corresponding to the given type used to look up the virtual machine.

Results

The operation returns the following results:

returnResult

This is the primary output, containing the confirmation message that the VM is undefined.

connectionUri

The connection URI that the RAS used to connect to the KVM host.

hostname

The DNS name of the KVM host.

4 Security

This section includes the following topics:

- About Linux KVM Security

About Linux KVM Security

For information on libvirt authentication and secure remote access, see:

- **http://libvirt.org/remote.html**
- **http://libvirt.org/auth.html**

The RAS needs an X.509 client certificate to connect using the TLS transport. To connect using the TLS transport, the RAS must have a client certificate which has been signed by the same CA as the server certificates of the KVM hosts to which it is to connect. Details on creating these certificates can be found on the libvirt.org site: http://libvirt.org/remote.html#Remote_certificates

Due to a bug in the libvirt java bindings, password authentication with SSH and TCP protocols are unsupported.

5 OO Tools

This section includes the following topic:

- OO Tools You Can Use with the Linux KVM-OO Integration

OO Tools You Can Use with the Linux KVM Integration

Following are OO tools that you can use with the Linux KVM integration:

- **RSFlowInvoke.exe and JRSFlowInvoke.jar**

  RSFlowInvoke (RSFlowInvoke.exe or the Java version, JRSFlowInvoke.jar) is a command-line utility that allows you to start a flow without using Central (although the Central service must be running). RSFlowInvoke is useful when you want to start a flow from an external system, such as a monitoring application that can use a command line to start a flow.

These tools are available in the Operations Orchestration home folder in /Studio/tools/.