Red Hat CloudForms 4.6

Methods Available for Automation

Advanced automation methods for Red Hat CloudForms

Last Updated: 2018-05-23

Red Hat CloudForms 4.6 Methods Available for Automation

Advanced automation methods for Red Hat CloudForms

Red Hat CloudForms Documentation Team
cloudforms-docs@redhat.com

Legal Notice

Abstract

This guide provides advanced methods for CloudForms Management Engine's automation functions. Methods can be used from within CloudForms Management Engine to create custom actions and workflows for the objects managed by CloudForms Management Engine. Information and procedures in this book are relevant to CloudForms Management Engine administrators. This document is organized by the object hierarchy in the Automate Model. If you have a suggestion for improving this guide or have found an error, please submit a Bugzilla report at

http://bugzilla.redhat.com against Red Hat CloudForms Management Engine for the Documentation component. Please provide specific details, such as the section number, guide name, and CloudForms version so we can easily locate the content.

Table of Contents

CHAPTER 1. METHODS AVAILABLE FOR USE WITH RED HAT CLOUDFORMS

Methods can be used from within Red Hat CloudForms to create custom actions and workflows for the objects managed for your Red Hat CloudForms Infrastructure. This document describes the methods available for use in Red Hat CloudForms. This document is organized by the object hierarchy in the Automate Model.

NOTE

Users of Red Hat CloudForms can construct custom automation methods in Ruby to extend the product. Red Hat CloudForms ships with a core set of Ruby gems used by the Red Hat CloudForms Rails Application. The Ruby gems in this set are subject to change, and have changed since the previous version. If you are calling gems using Automate that are no longer in the Red Hat CloudForms appliance, you can install them by using the **`gem install`** command.

While gems can be imported into automation methods using **`require`**, it is recommended that the authors of the automation methods clearly document the use of gems either in the core set or a custom set. It is the responsibility of the author of such custom automation to own the life cycle of any gem being referenced in those methods.

The Red Hat CloudForms 4.6 Release Notes , list Ruby gems that have been added, updated, or removed in the latest version of CloudForms Management Engine.

For lists of Ruby gems included in different CloudForms Management Engine releases, see the following article:

- https://access.redhat.com/articles/1534753

1.1. $EVM.ROOT

When an Automate method is launched, it has one global variable: **`$evm`**. The **`$evm`** variable allows the method to communicate back to Red Hat CloudForms. The **`$evm.root`** is the root object in the workspace, it provides access to the data currently loaded in the Red Hat CloudForms model. It use the objects data to solve more complex problems by integrating with Red Hat CloudForms methods.

The following is an excerpt from the **InspectMe** method that can be found in the **ManageIQ\System\Request** namespace. The dumpRoot method accesses the **`$evm.root`** object, and sends all of its attributes to the Red Hat CloudForms Automate log for review. In the dumpServer Method, the inspect method is run based on the value of the miq_server obtained from the **`$evm.root`** object.

```
############################
  #
  # Method: dumpRoot
  # Description: Dump Root information
  #
  ############################
  def dumpRoot
    $evm.log("info", "#{@log_prefix} - Root:<$evm.root> Begin Attributes")
    $evm.root.attributes.sort.each { |k, v| $evm.log("info", "#
{@log_prefix} - Root:<$evm.root> Attributes - #{k}: #{v}")}
    $evm.log("info", "#{@log_prefix} - Root:<$evm.root> End Attributes")
```

```
    $evm.log("info", "")
  end

  ##########################
  #
  # Method: dumpServer
  # Inputs: $evm.root['miq_server']
  # Description: Dump MIQ Server information
  #
  ##########################
  def dumpServer
    $evm.log("info","#{@log_prefix} - Server:<#
{$evm.root['miq_server'].name}> Begin Attributes")
    $evm.root['miq_server'].attributes.sort.each { |k, v| $evm.log("info",
"#{@log_prefix} - Server:<#{$evm.root['miq_server'].name}> Attributes - #
{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Server:<#
{$evm.root['miq_server'].name}> End Attributes")
    $evm.log("info", "")
  end
```

The result of dumpRoot is below. The value of miq_server is what gets passed into the dumpServer method.

```
[----] I, [2012-10-23T13:53:54.517279 #5320:f329024]  INFO -- : <User-
Defined Method> [InspectMe] - EVM Automate Method Started
[----] I, [2012-10-23T13:53:54.523637 #5320:f329024]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Begin Attributes
[----] I, [2012-10-23T13:53:54.527552 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes - miq_server: #
<MiqAeMethodService::MiqAeServiceMiqServer:0x0000001e76d900>
[----] I, [2012-10-23T13:53:54.528801 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes - miq_server_id:
1
[----] I, [2012-10-23T13:53:54.529961 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes - object_name:
Request
[----] I, [2012-10-23T13:53:54.531067 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes - request:
inspectme
[----] I, [2012-10-23T13:53:54.534054 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes - vm: DEV-JaneM
[----] I, [2012-10-23T13:53:54.535156 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes - vm_id: 85
[----] I, [2012-10-23T13:53:54.536238 #5320:ef8c538]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> Attributes -
vmdb_object_type: vm
[----] I, [2012-10-23T13:53:54.537159 #5320:f329024]  INFO -- : <User-
Defined Method> [InspectMe] - Root:<$evm.root> End Attributes
[----] I, [2012-10-23T13:53:54.537772 #5320:f329024]  INFO -- : <User-
Defined Method>
```

1.2. METHOD HIERARCHY

The Automate Model inline methods have a hierarchy. The sublevels in the hierarchy have access to the methods for itself and the levels above it. For example, Red Hat Hosts have access to the Red Hat Host methods, Host Methods, and Base Methods.

The following nested list displays the hierarchy.

- Top Level: Base
  - Authentication (authentication)
    - Private Keys (auth_private_key)
      - Key Pair for Clouds (auth_key_pair_cloud)
        - Amazon (auth_key_pair_amazon)
        - OpenStack (auth_key_pair_openstack)
  - Availability Zones (availability_zone)
    - Amazon (availability_zone_amazon)
    - OpenStack (availability_zone_openstack)
  - Classification (classification)
  - Cloud Networks (cloud_network)
  - Cloud Resource Quotas (cloud_resource_quota)
    - OpenStack (openstack_resource_quota)
  - Cloud Subnets (cloud_subnet)
  - Cloud Tenants (cloud_tenant)
  - Customization Templates (customization_template)
    - Cloud Init (customization_template_cloud_init)
    - Kickstart (customization_template_kickstart)
    - Sysprep (customization_template_sysprep)
  - Cluster (ems_cluster)
  - Event (ems_event)
  - Folder (ems_folder)
  - Providers (ext_management_system)
    - Cloud (ems_cloud)
      - Amazon (ems_amazon)
      - Openstack (ems_openstack)
    - Infrastructue (ems_infra)

        - Microsoft System Center VMM (ems_microsoft)
        - Red Hat Enterprise Virtualization (ems_redhat)
        - VMware vCenter (ems_vmware)
    - Filesystems (filesystem)
    - Firewall Rules (filewall_rule)
    - Flavors
        - Amazon (flavor_amazon)
        - OpenStack (flavor_openstack)
    - Floating IPs (floating_ip)
        - Amazon (floating_ip_amazon)
        - OpenStack (floating_ip_openstack)
    - Guest Applications (guest_application)
    - Guest Devices (guest_device)
    - Hardware (hardware)
    - Hosts (host)
        - Red Hat Enterprise Virtualization (host_redhat)
        - VMware (host_vmware)
            - VMware ESX (host_vmware_esx)
    - ISO Images (iso_image)
    - Jobs (job)
    - LANs (lan)
    - Groups (miq_group)
    - Policies (miq_policy)
    - Proxies (miq_proxy)
    - Requests (miq_request)
        - Automation (automation_request)
        - Host Provisioning (miq_host_provision_request)
        - VM Provisioning (miq_provision_request)
            - VM Templates (miq_provision_request_template)
        - Service Reconfiguration (service_reconfigure_request)

        - Service Template Provisioning (service_template_provision_request)
        - VM Migration (vm_migrate_request)
        - VM Reconfiguration (vm_reconfigure_request)
    - Request Task (miq_request_task)
        - Automation (automation_task)
        - Host Provisioning (miq_host_provision)
        - VM Provisioning (miq_provision)
            - Cloud (miq_provision_cloud)
                - Amazon (miq_provision_amazon)
                - OpenStack (miq_provision_openstack)
            - Red Hat Enterprise Virtualization (miq_provision_redhat)
                - Via ISO (miq_provision_redhat_via_iso)
                - Via PXE (miq_provision_redhat_via_pxe)
            - VMware (miq_provision_vmware)
                - Via NetApp RCU (miq_provision_vmware_via_net_app_rcu)
                - Via PXE (miq_provision_vmware_via_pxe)
        - Service Reconfiguration (service_reconfigure_task)
        - Service Template Provisioning (service_template_provision_task)
        - VM Migratation (vm_migrate_task)
        - VM Reconfiguration (vm_reconfigure_task)
    - Servers (miq_server)
    - Networks (network)
    - Operating Systems (operating_system)
    - PXE Images (pxe_image)
        - iPXE (pxe_image_ipxe)
        - PXELINUX (pxe_image_pxelinux)
    - PXE Servers (pxe_server)
    - Resource Pools (resource_pool)
    - Security Groups (security_group)
        - Amazon (security_group_amazon)

        - OpenStack (security_group_openstack)
    - Services (service)
    - Service Resources (service_resource)
    - Service Templates (service_template)
    - Snapshots (snapshot)
    - Storages (storage)
    - Switches (switch)
    - Users (user)
    - VMs or Templates (vm_or_template)
        - Templates (miq_template)
            - Cloud (template_cloud)
                - Amazon (template_amazon)
                - OpenStack (template_openstack)
            - Infrastructure (template_infra)
                - Microsoft (template_microsoft)
                - Red Hat Enterprise Virtualization (template_redhat)
                - VMware (template_vmware)
        - VMs (vm)
            - Clouds (vm_cloud)
                - Amazon (vm_amazon)
                - OpenStack (vm_openstack)
            - Infrastructure (vm_infra)
                - Microsoft (vm_microsoft)
                - Red Hat Enterprise Virtualization (vm_redhat)
                - Vmware (vm_vmware)
    - Windows Images (windows_images)

1.3. BASE METHODS

These methods may be used with all objects available in the Automate Model.

Method	Usage
inspect	Returns a string containing a list of attributes of the object. See the InspectMe method in Samples class
inspect_all	Returns all information for an object
virtual_column_names	Returns the objects virtual columns names
virtual_columns_inspect	Returns the objects virtual columns and values
reload	Returns to original object to prevent the internal object from being returned
model_suffix	Returns objects suffix. For an object of type MiqAeServiceVmVmware, returns "Vmware"
tagged_with?(category, name)	Is the object tagged with the category and name specified?
tags(category = nil)-- this means that category is an optional parameter, with a default of nil	Returns the tags.
tag_assign(tag)	Assigns tag to the object, except for the miq_provision object, which uses add_tag(category, tag_name)
tag_unassign(tag)	Unassigns tag to the object, except for the miq_provision object, which uses clear_tag(category, tag_name)

The **InspectMe Sample Method** uses many of the Methods shown in this document. The method returns attributes of the Red Hat CloudForms Server and then returns attributes for the host, cluster, and virtual machine from the provider of invocation. In many environments it is linked to a button.

```
#####################################
# EVM Automate Method: InspectMe
#
# Notes: Dump the objects in storage to the automation.log
#
#####################################
begin
  @method = 'InspectMe'
  @log_prefix = "[#{@method}]"
  $evm.log("info", "#{@log_prefix} - EVM Automate Method Started")

  # Turn on verbose logging
  @debug = true
```

```
  # List the types of object we will try to detect
  obj_types = %w{ vm host storage ems_cluster ext_management_system }
  obj_type = $evm.root.attributes.detect { |k,v| obj_types.include?(k)}

  # uncomment below to dump root object attributes
  dumpRoot

  # uncomment below to dump miq_server object attributes
  dumpServer


  # If obj_type is NOT nil
  unless obj_type.nil?
    rootobj = obj_type.first
    obj = obj_type.second
    $evm.log("info", "#{@log_prefix} - Detected Object:<#{rootobj}>")
    $evm.log("info","")

    case rootobj
    when 'host' then dumpHost(obj)
    when 'vm' then dumpVM(obj)
    when 'ems_cluster' then dumpCluster(obj)
    when 'ext_management_system' then dumpEMS(obj)
    when 'storage' then dumpStorage(obj)
    end
  end

  #
  # Exit method
  #
  $evm.log("info", "#{@log_prefix} - EVM Automate Method Ended")
  exit MIQ_OK

  #
  # Set Ruby rescue behavior
  #
rescue => err
  $evm.log("error", "#{@log_prefix} - [#{err}]\n#
{err.backtrace.join("\n")}")
  exit MIQ_ABORT
end
```

1.4. AVAILABILITY ZONES (AVAILABILITY_ZONE)

Method	Use
ext_management_system	Returns object's Management System
vms	Returns object's VMs
vms_and_templates	Returns object's VMs and templates

Method	Use
cloud_subnets	Returns object's cloud subnets

1.5. CLASSIFICATIONS (CLASSIFICATION)

Method	Use
parent	Returns object's parent object
namespace	Returns object's namespace
category	Returns object's category
name	Returns object's name
to_tag	Returns object's tag mapping

1.6. CLOUD NETWORKS (CLOUD_NETWORK)

Method	Use
ext_management_system	Returns object's Management System
cloud_tenant	Returns object's cloud tenant
cloud_subnets	Returns object's cloud subnets
security_groups	Returns object's security groups
vms	Returns object's VMs

1.7. CLOUD RESOURCE QUOTAS (CLOUD_RESOURCE_QUOTA)

Method	Use
ext_management_system	Returns object's Management System
cloud_tenant	Returns object's cloud tenant

1.8. CLOUD SUBNETS (CLOUD_SUBNET)

Method	Use
cloud_network	Returns object's cloud network
availability_zone	Returns object's availability zone
vms	Returns object's VMs

1.9. CLOUD TENANTS (CLOUD_TENANT)

Method	Use
ext_management_system	Returns object's Management System
security_groups	Returns object's security groups
cloud_networks	Returns object's cloud network
vms	Returns object's VMs
vms_and_templates	Returns object's VMs and templates
miq_templates	Returns object's templates
floating_ips	Returns object's floating IP addresses
cloud_resource_quotas	Returns object's quotas

1.10. CUSTOMIZATION TEMPLATE (CUSTOMIZATION_TEMPLATE)

Method	Use
Pxe_images	Returns customization templates pxe images

1.11. CLUSTERS (EMS_CLUSTER)

Method	Use
all_resource_pools	Return all of the objects Resource Pools
all_vms	Return all of the objects Virtual Machines
default_resource_pool	Return the objects default Resource Pool

Method	Use
ems_events	Returns an array of EmsEvent records associated with the object
ext_management_system	Return objects Management System
hosts	Return objects Hosts
parent_folder	Return objects Parent Folder
register_host(host)	Register Host to this Cluster
resource_pools	Return objects Resource Pools
storages	Return objects datastores
vms	Return objects Virtual Machines

```
  ############################
  #
  # Method: dumpCluster
  # Inputs: $evm.root['ems_cluster']
  # Description: Dump Cluster information
  #
  ############################
  def dumpCluster(cluster)
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> Begin
Attributes")
    cluster.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix}
- Cluster:<#{cluster.name}> Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> End
Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> Begin
Associations")
    cluster.associations.sort.each { |assc| $evm.log("info", "#
{@log_prefix} - Cluster:<#{cluster.name}> Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> End
Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> Begin
Virtual Columns")
    cluster.virtual_column_names.sort.each { |vcn| $evm.log("info", "#
{@log_prefix} - Cluster:<#{cluster.name}> Virtual Columns - #{vcn}: #
{cluster.send(vcn)}")}
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> End
```

```
Virtual Columns")
    $evm.log("info","")
  end
```

1.12. MANAGEMENT SYSTEM EVENTS (EMS_EVENT)

Method	Use
ext_management_system	Returns object's provider
ems	Shortcut to ext_management_system
src_vm	Source VM for the event
vm	VM for the event
src_host	Source Host for the event
host	Host for the event
dest_vm	Destination VM for the event
service	Service for the event
dest_host	Destination Host for the event
refresh(*targets)	Refresh the target types specified (ems, vm, host, src_vm, src_host, dest_vm, or dest_host)

1.13. MANAGEMENT SYSTEM FOLDERS (EMS_FOLDER)

Method	Use
hosts	Returns hosts that are in the folder
vms	Returns VMs that are in folder
register_host(host)	Registers specified host to the folder
folder_path(*options)	Returns folders path

1.14. PROVIDERS (EXT_MANAGEMENT_SYSTEM)

Method	Use
authentication_password_encrypted	Returns credentials password encrypted
authentication_password	Returns credentials password unencrypted
authentication_userid	Returns credentials user id
ems_clusters	Returns objects clusters
ems_events	Returns an array of EmsEvent records associated with the object
ems_folders	Returns objects folders
hosts	Returns objects hosts
refresh	Refreshes relationships and power states for objects related to the object
resource_pools	Returns objects resource pools
storages	Returns objects storages
vms	Returns objects vms
to_s	Converts object to string

```
  ############################
  #
  # Method: dumpEMS
  # Inputs: $evm.root['ext_management_system']
  # Description: Dump EMS information
  #
  ############################
  def dumpEMS(ems)
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin Attributes")
    ems.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} -
EMS:<#{ems.name}> Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin
Associations")
    ems.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} -
EMS:<#{ems.name}> Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin EMS
```

```
Folders")
    ems.ems_folders.each { |ef| ef.attributes.sort.each { |k,v|
$evm.log("info", "#{@log_prefix} - EMS:<#{ems.name}> EMS Folder:<#
{ef.name}> #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End EMS Folders")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin Virtual
Columns")
    ems.virtual_column_names.sort.each { |vcn| $evm.log("info", "#
{@log_prefix} - EMS:<#{ems.name}> Virtual Columns - #{vcn}: #
{ems.send(vcn)}")}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End Virtual
Columns")
    $evm.log("info","")
  end
```

1.14.1. Cloud Providers (ems_cloud)

Method	Use
availability_zones	Return the provider's availability zones
cloud_networks	Return the provider's available networks
cloud_tenants	Return the provider's available tenants
flavors	Return the provider's hardware flavors
floating_ips	Return the provider's floating IP addresses
key_pairs	Return the provider's key pairs
security_groups	Return the provider's security groups
cloud_resource_quotas	Return the provider's resource quotas

1.15. FIREWALL RULES (FIREWALL_RULE)

Method	Use
resource	Return object's resource
source_security_group	Return object's source security group

1.16. FLAVORS (FLAVOR)

Method	Use
ext_management_system	Returns object's Management System
vms	Returns object's VMs

1.17. FLOATING IPS (FLOATING_IP)

Method	Use
ext_management_system	Returns object's Management System
vm	Returns object's VMs
cloud_tenant	Returns object's cloud tenant

1.18. GUEST APPLICATIONS (GUEST_APPLICATION)

Method	Use
vm	Returns objects VM
host	Returns objects Host

1.19. GUEST DEVICE (GUEST_DEVICE)

Method	Use
hardware	Returns objects hardware
switch	Returns objects switch
lan	Returns objects LAN
network	Returns objects network

1.20. HARDWARE (HARDWARE)

Method	Use
ipaddresses	Returns objects IP addresses

Method	Use
guest_devices	Returns objects guest devices
storage_adapters	Returns objects storage adapters
nics	Returns objects nics
ports	Returns objects ports
vm	Returns objects Virtual Machine
host	Returns objects Host
mac_addresses	Returns objects MAC addresses

1.21. HOSTS (HOST)

Method	Use
authentication_password	Returns credential password
authentication_userid	Returns credential user
datacenter	Returns datacenter
directories	Returns list of directories for the object
domain	Returns the domain portion of the hostname
ems_cluster	Returns cluster
ems_events	Returns an array of EmsEvent records associated with the object
ems_folder	Returns hosts folder on Management System

Method	Use
event_log_threshold?(options)	Searches event log records to determine if an event has occurred x number of times within a defined time frame. Returns true if the number of matching records found are greater or equal to the specified freq_threshold, otherwise it returns false Options values: :message_filter_type - Must be one of "STARTS WITH", "ENDS WITH", "INCLUDES", "REGULAR EXPRESSION" :message_filter_value - <string value to search for> :time_threshold - Options time interval to search. Example: 2.days (Search the past 2 days of event logs) Default 10.days :freq_threshold - Number of occurrences to check for. Default = 2 :source, :event_id, :level, :name - Options filter values
ext_management_system	Returns Management System
files	Returns list of files for the object
guest_applications	Returns Guest Applications
hardware	Returns hardware
lans	Returns LANs
operating_system	Returns Operating System
storages	Returns datastores
switches	Returns network switches
vms	Returns VMs.

Method	Use
credentials(type = :remote)	Returns credentials for a Host for the specified type as an array for username/pwd. (Default type is :remote if no type is specified.) Supports 4 different types of credentials: :default = Default :remote = Remote Login (think SSH for ESX) :ws = Web Services :ipmi = IPMI Example 1: cred = host.credentials cred ⇒ ["user", "pwd"] Example 2: user_str, pwd_str = host.credentials(:ipmi) user_str ⇒ "user" pwd_str ⇒ "pwd"
ems_custom_keys	Returns Management Systems custom keys
ems_custom_get(key)	Gets Value for specified Management Systems custom key
ems_custom_set(attribute, value)	Sets value for specified custom key of the Management System
custom_keys	Lists Red Hat CloudForms Server custom keys
custom_get(key)	Gets value for specified Red Hat CloudForms Server custom key
custom_set(key, value)	Sets value for specified Red Hat CloudForms Server custom key
ssh_exec(script)	Runs the specified script on the host
get_realtime_metric(metric, range, function)	Returns specified realtime metric
current_memory_usage	Returns current memory usage

Method	Use
current_cpu_usage	Returns current cpu usage
current_memory_headroom	Returns current memory headroom
to_s	Converts object to string
scan	Performs SmartState Analysis on the object

The following table lists the metric types available for the **`get_realtime_metric(metric, range, function)`** method for hosts.

Metric	Description
v_derived_storage_used	Capacity - Used space in bytes
v_pct_cpu_ready_delta_summation	CPU - Percentage ready
v_pct_cpu_wait_delta_summation	CPU - Percentage wait
v_pct_cpu_used_delta_summation	CPU - Percentage used
v_derived_host_count	State - Number of hosts (Hourly Count / Daily Average)
v_derived_cpu_reserved_pct	CPU - Percentage available
v_derived_memory_reserved_pct	Memory - Percentage available

The following Ruby snippet demonstrates using the **`get_realtime_metric(metric, range, function)`** method using the **`v_pct_cpu_ready_delta_summation`** metric.

```
host = $evm.root['host']
cpu_rdy = host.get_realtime_metric(:v_pct_cpu_ready_delta_summation,
[15.minutes.ago.utc,5.minutes.ago.utc], :avg)
```

```
############################
  #
  # Method: dumpHost
  # Inputs: $evm.root['host']
  # Description: Dump Host information
  #
  ############################
  def dumpHost(host)
    host = $evm.object['host'] || $evm.root['host']
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin
Attributes")
```

```
    host.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} -
Host:<#{host.name}> Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin
Associations")
    host.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix}
- Host:<#{host.name}> Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End
Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin
Hardware")
    host.hardware.attributes.each { |k,v| $evm.log("info", "#{@log_prefix}
- Host:<#{host.name}> Hardware - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Hardware")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Lans")
    host.lans.each { |lan| lan.attributes.sort.each { |k,v|
$evm.log("info", "#{@log_prefix} - Host:<#{host.name}> Lan:<#{lan.name}> -
#{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Lans")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin
Switches")
    host.switches.each { |switch| switch.attributes.sort.each { |k,v|
$evm.log("info", "#{@log_prefix} - Host:<#{host.name}> Swtiche:<#
{switch.name}> - #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Switches")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Operating
System")
    host.operating_system.attributes.sort.each { |k, v| $evm.log("info",
"#{@log_prefix} - Host:<#{host.name}> Operating System - #{k}: #
{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Operating
System")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Guest
Applications")
    host.guest_applications.each { |guest_app|
guest_app.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} -
Host:<#{host.name}> Guest Application:<#{guest_app.name}> - #{k}: #
{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Guest
Applications")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Virtual
Columns")
```

```
    host.virtual_column_names.sort.each { |vcn| $evm.log("info", "#
{@log_prefix} - Host:<#{host.name}> Virtual Columns - #{vcn}: #
{host.send(vcn).inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Virtual
Columns")
    $evm.log("info", "")
  end
```

1.21.1. Hosts: Vmware ESX (host_vmware_esx)

Method	Use
disable_vmotion(device = nil)	Disable vMotion
enable_vmotion(device = nil)	Enable vMotion
enter_maintenance_mode(timeout = 0, evacuate = false)	Put Host in Maintenance Mode
exit_maintenance_mode(timeout = 0)	Leave Maintenance Mode
in_maintenance_mode?	Check to see if the host is in Maintenance Mode
power_down_to_standby(timeout = 0, evacuate = false)	Put Host in standby
power_up_from_standby(timeout = 0)	Take Host out of standby
reboot(force = false)	Reboot Host
shutdown(force = false)	Shutdown Host
vmotion_enabled?(device = nil)	Check to see if vMotion is enabled

1.22. LAN (LAN)

Method	Use
switch	Returns objects switch
guest_devices	Returns objects guest devices
vms	Returns objects Virtual Machines
templates	Returns objects templates
hosts	Returns objects Hosts

1.23. GROUPS (MIQ_GROUP)

Method	Use
users	Returns users in the current miq_group
vms	Returns Virtual Machines that this group owns
custom_keys	Returns all custom keys for the group
custom_get(key)	Returns the value of the specified custom key for the group
custom_set(key, value)	Sets the value for the specified key

```
  ############################
  #
  # Method: dumpGroup
  # Inputs: $evm.root['user'].miq_group
  # Description: Dump User's Group information
  #
  ############################
  def dumpGroup
    user = $evm.root['user']
    unless user.nil?
      miq_group = user.miq_group
      unless miq_group.nil?
        $evm.log("info","#{@method} - Group:<#{miq_group.description}>
Begin Attributes [miq_group.attributes]")
        miq_group.attributes.sort.each { |k, v| $evm.log("info", "#
{@method} - Group:<#{miq_group.description}> Attributes - #{k}: #
{v.inspect}")} unless $evm.root['user'].miq_group.nil?
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> End
Attributes [miq_group.attributes]")
        $evm.log("info", "")

        $evm.log("info","#{@method} - Group:<#{miq_group.description}>
Begin Associations [miq_group.associations]")
        miq_group.associations.sort.each { |assc| $evm.log("info", "#
{@method} - Group:<#{miq_group.description}> Associations - #{assc}")}
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> End
Associations [miq_group.associations]")
        $evm.log("info","")

        $evm.log("info","#{@method} - Group:<#{miq_group.description}>
Begin Virtual Columns [miq_group.virtual_column_names]")
        miq_group.virtual_column_names.sort.each { |vcn| $evm.log("info",
"#{@method} - Group:<#{miq_group.description}> Virtual Columns - #{vcn}: #
{miq_group.send(vcn).inspect}")}
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> End
Virtual Columns [miq_group.virtual_column_names]")
        $evm.log("info","")
```

```
        end
      end
    end
```

1.24. REQUEST (MIQ_REQUEST)

Request objects are submitted to Red Hat CloudForms Server for processing. After the request phase, the request becomes a task object. The table below shows the relationship between a request object and a task object.

Request Object	Task Object
automation_request	automation_task
miq_host_provision_request	miq_host_provision
miq_provision_request	miq_provision
vm_reconfigure_request	vm_reconfigure_task
service_template_provision_request	service_template_provision_task
vm_migrate_request	vm_migrate_task

If you set something on the request object, it will be inherited by the task instance that does the work. This may be useful if you are provisioning multiple virtual machines at a time and need to modify the same setting for all. Otherwise, the item can be modified on the individual task.

Method	Use
add_tag(category, tag_name)	Add a tag to a provision instance by specifying the category and tag name Example: miq_provision.add_tag(:location, "CHI")
approve(approver,reason for approval)	Approves the miq_request. Example: $evm.root["miq_request"].approve("admin", "Auto-Approved")
approvers	Returns request approvers.
authorized?	Returns true if authorized, false if not.

Method	Use
clear_tag(category=nil, tag_name=nil)	Without any parameters, the clear_tag method will clear all tags from the provision request. Providing a category will clear all tags selected in that category. Clear a specific category/tag by providing it. Example: miq_provision.clear_tag(:location, "CHI")
deny	Denies the miq_request Example: # Deny the request $evm.log('info',"Request denied because of Quota") $evm.root["miq_request"].deny("admin", "Quota Exceeded")
get_classification(category)	Works the same as get_tag(category) but the returned data is a hash with :name and :description Example: request.get_classification(:department) Returns: [{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]
get_classifications	Works the same as get_tag but the returned tag data is a hash with :name and :description Example: request.get_classifications Returns: {:cc⇒{:name⇒"001", :description⇒"Cost Center 001"}, :department⇒[{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]}

Method	Use
get_option(key)	Returns a value from the options hash based on the name of the key name passed in. Internally many of the values are stored as an array of items. (For example, a target host would be stored as the index to the host object in the db and the display name.) Calling this method will return the first item if it is an array. For non-array values the item is returned unmodified. Example: miq_provision_request.get_option(:number_of_cpus)
get_tag(category)	Returns the tags selected for the specified tag category. Example: request.get_tag(:department) Returns: ["accounting", "engineering"]
get_tags	Get all selected tags stored in a hash by category. If more than one tag is selected in a category, the hash will contain an array of tag names. Otherwise it will contain the tag name as a string. Example: request.get_tags Returns: {:cc⇒"001", :department⇒["accounting", "engineering"]}
miq_request	(Legacy support) Internal Note: The miq_request instance use to be a separate instance from the specific request instance (like miq_provision_request). When the classes were refactored into 1 this method was added to allow existing code and automate methods to continue to run unchanged.)
miq_request_tasks	Returns the requests tasks
options	Returns a hash containing all the options set for the current provision object
pending	Puts the object in a pending state for approval Example: # Raise automation event: request_pending $evm.root["miq_request"].pending

Method	Use
reason	Returns reason for approval or denial of request
requester	Returns the requester
resource	Returns the resource for the request
set_message(value)	Sets the message for the request
set_option(key, value)	Sets the specified key/value pair for the object

1.24.1. Automation Request (automation_request)

Method	Use
automation_tasks	Returns objects automate tasks

1.24.2. Host Provision Request (miq_host_provision_request)

Method	Use
miq_host_provisions	Returns the miq_host_provisions objects
ci_type	Returns the cloud infrastructure type: host

1.24.3. VM Provision Request (miq_provision_request)

Method	Use
check_quota(quota_type, options={})	Returns the quota information for the specified type
ci_type	Returns the cloud infrastructure type: vm
eligible_clusters	Returns an array of available Cluster objects filtered by previously selected resources
eligible_customization_templates	Returns an array of available Customization Templates filtered by previously selected resources
eligible_folders	Returns an array of available Folder objects filtered by previously selected resources
eligible_hosts	Returns an array of available Host objects filtered by previously selected resources

Method	Use
eligible_iso_images	Returns an array of available ISO image objects filtered by previously selected resources
eligible_pxe_images	Returns an array of available PXE Image objects filtered by previously selected resources
eligible_pxe_servers	Returns an array of available PXE Server objects filtered by previously selected resources
eligible_resource_pools	Returns an array of available Resource Pool objects filtered by previously selected resources
eligible_resources(rsc_type)	Returns eligible resources given the type specified
eligible_storages	Returns an array of available Storage (Datastore) objects filtered by previously selected resources
eligible_windows_images	Returns an array of available Windows Image objects filtered by previously selected resources
get_folder_paths	Returns a hash where the key is an index and the value is the fully-qualified path name of the folder. (Sample: {7 ⇒ Dev/Dept1/QA, 8 ⇒ Test/Dept2/QA}) This format is useful when a fully-qualified path is required to match the folder name. For example, if you had multiple QA folders under different departments in the sample above. To find the proper QA folder you need to evaluate the entire folder path.
get_retirement_days	Returns the number of dates until retirement
miq_provision	Returns the task.
miq_request	Returns the miq_provision_requests miq_request object
set_cluster(rsc)	Set the cluster to use based on object returned from eligible_clusters
set_customization_template(rsc)	Set the customization_template to use based on object returned from eligible_customization_templates

Method	Use
set_folder(folder_path)	Set the folder to use based on object returned from eligible_folders. In addition, set_folder accepts the following folder types: Folder Paths - separated by forward slashes. (Must include Data-center name.) Example: Prod/Discovered virtual machine where Prod is the Data-center name and Discovered virtual machine is the folder name. Object returned from the get_folder_paths method
set_host(rsc)	Set the host to use based on object returned from eligible_hosts
set_network_adapter(idx, nic_hash, value=nil)	Modifies the network card attached to the VM container Available settings: :is_dvs true / false (Default: false) :network (Network Name) :mac_address :devicetype (Default: VirtualPCNet32) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.device.VirtualEthernetCard.html :connectable ⇒ {:allowguestcontrol ⇒ true / false} (Default: true) :connectable ⇒ {:startconnected ⇒ true / false} (Default: true) :connectable ⇒ {:connected ⇒ true / false} (Default: true) Example: prov.set_network_adapter(1, {:network ⇒ dvs_net1, :is_dvs ⇒ true})
set_network_address_mode(mode)	Sets IP address type. Available modes are dhcp and static

Method	Use
set_nic_settings(idx, nic_hash, value=nil)	Modifies the network interface settings at the operating system level Available settings: :addr_mode "dhcp" / "static" (Default: static) :ip_addr :subnet_mask :gateway :dns_domain :dns_servers (Windows Only) Comma separated values :sysprep_netbios_mode (Windows Only) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.customization.IPSettings.NetBIOSMode.html :wins_servers Passed as a string specifying the Primary and Secondary WINS servers separated by a comma. "<PrimaryWINS>, <SecondaryWINS>" Example: prov.set_nic_settings(1, {:ip_addr⇒10.226.133.55, :subnet_mask⇒'255.255.255.192', :gateway⇒'10.226.133.5', :addr_mode⇒["static", "Static"] })
set_iso_image(rsc)	Set the iso_image to use based on object returned from eligible_iso_images
set_pxe_image(rsc)	Set the pxe_image to use based on object returned from eligible_pxe_images
set_pxe_server(rsc)	Set the pxe_server to use based on object returned from eligible_pxe_servers
set_resource_pool(rsc)	Set the resource_pool to use based on object returned from eligible_resource_pools
set_resource(rsc)	Sets the resource for the request. (Helper method, should not be called directly)
set_retirement_days	Set the number of days until retirement.

Method	Use
set_storage(rsc)	Set the Datastore (storage object) to use based on object returned from eligible_storages
set_vm_notes(note)	Sets text for the VM notes (aka annotation) field
set_windows_image(rsc)	Set the windows_image to use based on object returned from eligible_windows_images
source_type	Returns the provision source type. (values are 'vm' or 'template')
src_vm_id	Returns ID of the template being cloned
target_type	Returns the provision target type. (values are 'vm' or 'template')
vm_template	Returns the requests template

1.25. REQUEST TASK (MIQ_REQUEST_TASK)

Method	Use
add_tag(category, tag_name)	Add a tag to a provision instance by specifying the category and tag name. Example: miq_provision.add_tag(:location, "CHI")
clear_tag(category=nil, tag_name=nil)	Without any parameters, the clear_tag method will clear all tags from the provision request. Providing a category will clear all tags selected in that category. Clear a specific category/tag by providing it. Example: miq_provision.clear_tag(:location, "CHI")
destination	Returns the destination object. (The resultant object from running the task. In the case of provisioning, this would be the newly created VM.)
execute	Executes or processes the request.
finished(msg)	Sets the task to finished with the supplied message.

Method	Use
get_classification(category)	Works the same as get_tag(category) but the returned data is a hash with :name and :description. Example: request.get_classification(:department) Returns: [{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]
get_classifications	Works the same as get_tag but the returned tag data is a hash with :name and :description. Example: request.get_classifications Returns: {:cc⇒{:name⇒"001", :description⇒"Cost Center 001"}, :department⇒[{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]}
get_option_last(key)	This method is the same as get_option, except that it returns the last array value.
get_option(key)	Returns a value from the options hash based on the name of the key name passed in. Internally many of the values are stored as an array of items. (For example, a target host would be stored as the index to the host object in the db and the display name.) Calling this method will return the first item if it is an array. For non-array values the item is returned unmodified. Example: miq_provision_request.get_option(:number_of_cpus)
get_tag(category)	Returns the tags selected for the specified tag category. Example: request.get_tag(:department) Returns: ["accounting", "engineering"]

Method	Use
get_tags	Get all selected tags stored in a hash by category. If more than one tag is selected in a category, the hash will contain an array of tag names. Otherwise it will contain the tag name as a string. Example: request.get_tag Returns: {:cc⇒"001", :department⇒["accounting", "engineering"]}
message=(msg)	Sets the message for the request task.
miq_request	Returns the miq_request for the task.
miq_request_task	Returns the parent miq_request task.
miq_request.user_message=(msg)	Allows automate method to override the default request status message. As a result, you can set custom messages for the progression of the state machine and its success or failure. This message will appear as the last message for the service request.
miq_request_tasks	Returns the children miq_request tasks.
options	Returns a hash containing all the options set for the current object.
set_option(key, value)	Updates a key/value pair in the options hash for the provision object. Often the value is required to be an array.
source	Returns the source object. (The source, or input, object that the task runs against. In the case of provisioning, this would be the VM or template selected to be provisioned.)

1.25.1. Automation Task (automation_task)

Method	Use
automation_request	Returns associated automation_request object
status	Returns status of the task

1.25.2. Host Provision Task (miq_host_provision)

Method	Use
host	Returns objects host
miq_host_provision_request	Returns the request that created the task
status	Returns status of host provision

1.25.3. VM Provision Task (miq_provision)

Method	Use
check_quota(quota_type, options={})	Returns the quota information for the specified type
eligible_clusters	Returns an array of available Cluster objects filtered by previously selected resources
eligible_customization_templates	Returns an array of available Customization Templates
eligible_folders	Returns an array of available Folder objects filtered by previously selected resources
eligible_hosts	Returns an array of available Host objects filtered by previously selected resources
eligible_iso_images	Returns an array of available ISO Image objects filtered by previously selected resources
eligible_pxe_images	Returns an array of available PXE Image objects filtered by previously selected resources
eligible_pxe_servers	Returns an array of available PXE Servers filtered by previously selected resources
eligible_resource_pools	Returns an array of available Resource Pool objects filtered by previously selected resources
eligible_resources(rsc_type)	Returns the eligible resources for the resource type specified
eligible_storages	Returns an array of available Storage (Datastore) objects filtered by previously selected resources
eligible_windows_images	Returns an array of available Windows Image objects filtered by previously selected resources

Method	Use
get_domain_details	Returns domain information
get_domain_name	Returns domain name
get_folder_paths	Returns a hash where the key is an index and the value is the fully-qualified path name of the folder. (Sample: {7 ⇒ Dev/Dept1/QA, 8 ⇒ Test/Dept2/QA}) This format is useful when a fully-qualified path is required to match the folder name. For example, if you had multiple QA folders under different departments in the sample above. To find the proper QA folder you need to evaluate the entire folder path.
get_network_details	Returns network information
get_network_scope	Returns network scope
miq_provision_request	Returns the provision request object
set_cluster(rsc)	Set the cluster to use based on object returned from eligible_clusters
set_customization_spec(name=nil, override=false)	Sets the name of the custom spec to use as defined by its name in Virtual Center
set_customization_template(rsc)	Set the customization_template to use based on object returned from eligible_customization_templates
set_dvs(portgroup, switch = portgroup)	Set the name of the Distributed Virtual Switch (portgroup). An options <switch> name can also be passed Example: miq_provision.set_dvs('default')
set_folder(folder_path)	Set the folder to use based on object returned from eligible_folders. In addition, set_folder accepts the following folder types: Folder Paths - separated by forward slashes. (Must include Data-center name.) Example: Prod/Discovered virtual machine where Prod is the Data-center name and Discovered virtual machine is the folder name. Object returned from the get_folder_paths method

Method	Use
set_host(rsc)	Set the host to use based on object returned from eligible_hosts
set_network_adapter(idx, nic_hash, value=nil)	Modifies the network card attached to the VM container Available settings: :is_dvs true / false (Default: false) :network (Network Name) :mac_address :devicetype (Default: VirtualPCNet32) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.device.VirtualEthernetCard.html :connectable ⇒ {:allowguestcontrol ⇒ true / false} (Default: true) :connectable ⇒ {:startconnected ⇒ true / false} (Default: true) :connectable ⇒ {:connected ⇒ true / false} (Default: true) Example: prov.set_network_adapter(1, {:network ⇒ dvs_net1, :is_dvs ⇒ true})
set_network_address_mode(mode)	Available modes are dhcp and static

Method	Use
set_nic_settings(idx, nic_hash, value=nil)	Modifies the network interface settings at the operating system level Available settings: :addr_mode "dhcp" / "static" (Default: Statis) :ip_addr :subnet_mask :gateway :dns_domain :dns_servers (Windows Only) Comma separated values :sysprep_netbios_mode (Windows Only) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.customization.IPSettings.NetBIOSMode.html :wins_servers Passed as a string specifying the Primary and Secondary WINS servers separated by a comma. "<PrimaryWINS>, <SecondaryWINS>" Example: prov.set_nic_settings(1, {:ip_addr⇒10.226.133.55, :subnet_mask⇒'255.255.255.192', :gateway⇒'10.226.133.5', :addr_mode⇒["static", "Static"] })
set_iso_image(rsc)	Set the iso_image to use based on object returned from eligible_iso_images
set_pxe_image(rsc)	Set the pxe_image to use based on object returned from eligible_pxe_images
set_pxe_server(rsc)	Set the pxe_server to use based on object returned from eligible_pxe_servers
set_resource_pool(rsc)	Set the resource_pool to use based on object returned from eligible_resource_pools
set_storage(rsc)	Set the Datastore (storage object) to use based on object returned from eligible_storages
set_vlan(vlan)	Sets the name of the VLan to use Example: miq_provision.set_vlan('default')

Method	Use
set_vm_notes(note)	Sets text for the VM notes (aka annotation) field
set_vm_notes(notes)	Sets text for the VM notes (aka annotation) field
set_windows_image(rsc)	Set the windows_image to use based on object returned from eligible_windows_images
source_type	Returns the provision source type. (values are 'vm' or 'template')
status	Returns provision status
target_type	Returns the provision target type. (values are 'vm' or 'template')
vdi_farm	Returns VDI Farm information
vm	The newly created vm
vm_template	Returns the template selected to be provisioned

```
begin
  miq_provision = $evm.root["miq_provision"] || $evm.root['miq_provision']
  prov = $evm.root["miq_provision"]
  user = prov.miq_request.requester
  raise "User not specified" if user.nil?



  ####################################
  # Process Change Request Number and set VM Annotation
  ####################################
  intake = prov.get_option(:vm_description)
  intake = "Change Request#: #{intake}"
  prov.set_option(:vm_description,intake)


  ####################################
  # Set the customization spec based on the environment tag chosen in the
dialog
  ####################################
  tags = prov.get_tags
  $evm.log("info","Tags: #{tags.inspect}")
  env = tags[:environment]
  $evm.log("info", "Mapping custom spec based on Category Environment <#
{env}> chosen in the dialog")
  if env.eql? "dev"
    customization_spec = "Dev-Specification"
    miq_provision.set_customization_spec(customization_spec)
```

```
  end
  if env.eql? "stg"
    customization_spec = "Stg-Specification"
    miq_provision.set_customization_spec(customization_spec)
  end


  ###################################
  # Set the VM Notes as follows:
  ###################################
  vm_notes = "#{intake}"
  vm_notes +=  "\nOwner: #{miq_provision.get_option(:owner_first_name)} #
{miq_provision.get_option(:owner_last_name)}"
  vm_notes += "\nEmail: #{miq_provision.get_option(:owner_email)}"
  vm_notes += "\nSource Template: #{miq_provision.vm_template.name}"
  miq_provision.set_vm_notes(vm_notes)


  ###################################
  # Drop the VM in the targeted folder
  # In VC a folder must exist that matches the LDAP Group
  # VM will be placed in the Folder
  ###################################
  if prov.get_option(:placement_folder_name).nil?
    ###################################
    # If you want to use a Default folder, set folder = below to the
default
    ###################################
    #    folder = "22F DC/LAB
FARM/GSE/Intel/Infrastructure/ManageIQ/SelfServiceVMs"
    folder = "DC1/Infrastructure/ManageIQ/SelfService"
    $evm.log("info", "Placing VM in VC folder: <#{folder}")
    $evm.log("info", "Set_folder called with [#{folder.inspect}]")

    miq_provision.set_folder(folder)
  end


  ########################################################
  # Set the IP Address based on the :mac_address entered in the dialog
  #
  ########################################################
  ipaddr = prov.get_option(:mac_address)

  if ! ipaddr.nil?
    # Set provisioning options to override options
    prov.set_option(:sysprep_spec_override, [true, 1])
    prov.set_option(:addr_mode, ["static", "Static"])
    prov.set_option(:ip_addr, ipaddr)
    # Reset :mac_address to nil
    prov.set_option(:mac_address, nil)
  end

  $evm.log("info", "Provision Options: #{prov.options.inspect}")

  exit MIQ_OK
```

```
rescue => err
  $evm.log("info", "Set_folder err [#{err}]\n#{err.backtrace.join("\n")}")
end
```

1.25.3.1. Adding Disks to OpenStack Instance Provisioning

Using the clone_options, construct the keys and values required to define new disks as well as override or set other keys if needed.

Example 1.1. Usage in Automate Model

Include the following settings with prov.set_options to set one or multiple volumes to attach. Note that each hash requires the :volume_id and :device_name keys.

```
clone_options = {
  :block_device_mapping => [
    {:volume_id => "d4xxxx92-5xx0-4xx9-bxx4-f1xxxxxxxx19", :device_name =>
"/dev/sdb"},
    {:volume_id => "e7xxxxf5-dxxd-4xx0-8xx3-a2xxxxxxxx12", :device_name =>
"/dev/sdc"},
  ]
}

prov = $evm.root["miq_provision"]
prov.set_option(:clone_options, clone_options)
```

1.25.3.2. Booting OpenStack Instances from Volume

You can boot instances created in Red Hat Enterprise Linux OpenStack Platform from a specified volume from an existing selection of block volumes. You can enable automate to pass the required parameters to OpenStack to use volumes as booting devices on an instance.

Example 1.2. Usage in Automate Model:

Include the following settings in the prov.set_option for provisioning instances created in OpenStack.

```
prov.set_option(
  :clone_options, {
    :image_ref => nil,
    :block_device_mapping_v2 => [{
      :boot_index => 0,
      :uuid => "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx",
      :device_name => "vda",
      :source_type => "volume",
      :destination_type => "volume",
      :delete_on_termination => false
    }]
  }
)
```

> You can allow passing "block_device_mapping_v2" in the clone options and remove keys from clone_options by setting value to nil.

1.25.3.3. VM Provisioning for Clouds Task (miq_provision_cloud)

Method	Use
availability_zones	Returns object's availability zones
instance_types	Returns object's instance types
security_groups	Returns object's security groups
floating_ip_addresses	Returns object's floating IP addresses
cloud_networks	Returns object's cloud network
cloud_subnets	Returns object's cloud subnet
guest_access_key_pairs	Returns object's guest key pair
cloud_tenants	Returns object's cloud tenant

1.25.4. Service Template Provision Task (service_template_provision_task)

Method	Use
dialog_options	Returns objects dialog options hash
dialog_parser	Provides consolidated and configurable dialog parsing
get_dialog_option(key)	Returns objects dialog value for the specified key
service_resource	Returns the service resource for the task
set_dialog_option(key, value)	Sets a dialog option Example: set_dialog_option('memory',memory_size)
status	Returns the tasks status

1.25.5. Service Reconfiguration Task (service_reconfigure_task)

Method	Use
dialog_options	Show all dialog options for object
get_dialog_option(key)	Show a dialog option based stored in key
set_dialog_option(key, value)	Set value as a dialog option in key
status	Returns status of the task
finished(msg)	Sets the task to finished with the supplied message

1.25.6. VM Migrate Task (vm_migrate_task)

Method	Use
status	Returns status of the migration task

1.26. PROXIES (MIQ_PROXY)

Method	Use
host	Returns object's hosts
powershell(script, returns = 'string')	Submits a powershell script

1.27. SERVERS (MIQ_SERVER)

These methods are available to the Red Hat CloudForms Server.

Method	Use
zone	Returns Red Hat CloudForms Servers Zone
region_number	Returns Red Hat CloudForms Servers Region Number
region_name	Returns Red Hat CloudForms Servers Region Name

```
##########################
#
# Method: dumpServer
# Inputs: $evm.root['miq_server']
# Description: Dump MIQ Server information
#
```

```
  ############################
  def dumpServer
    $evm.log("info","#{@method} - Server:<#{$evm.root['miq_server'].name}>
Begin Attributes")
    $evm.root['miq_server'].attributes.sort.each { |k, v| $evm.log("info",
"#{@method} - Server:<#{$evm.root['miq_server'].name}> Attributes - #{k}:
#{v.inspect}")}
    $evm.log("info","#{@method} - Server:<#{$evm.root['miq_server'].name}>
End Attributes")
    $evm.log("info", "")
  end
```

1.28. NETWORK (NETWORK)

Method	Use
hardware	Returns objects hardware
guest_device	Returns objects guest devices

1.29. PXE IMAGE (PXE_IMAGE)

Method	Use
customization_templates	Returns objects customization templates
pxe_server	Returns objects pxe server

1.30. PXE SERVER (PXE_SERVER)

Method	Use
advertised_images	Returns objects advertised images
advertised_pxe_images	Returns objects advertised pxe images
default_pxe_image_for_windows	Returns objects default pxe image for windows
discovered_images	Returns objects discovered images
discovered_pxe_images	Returns objects discovered pxe images
images	Returns objects images
pxe_images	Returns objects pxe_images

Method	Use
windows_images	Returns objects windows images

1.31. SECURITY GROUPS (SECURITY_GROUP)

Method	Use
ext_management_system	Returns object's Management System
cloud_network	Returns object's cloud network
cloud_tenant	Returns object's cloud tenant
firewall_rules	Returns object's firewall rules
vms	Returns object's VMs

1.32. SERVICE (SERVICE)

Method	Use
custom_keys	Returns custom keys
custom_get(key)	Gets value for specified custom key
custom_set(attribute, value)	Sets value for specified custom key
display=(display)	Set display option
group=(group)	Sets group that owns the service
name=(new_name)	Sets name of service
owner=(owner)	Sets owner of the service
retire_now	Retire Service immediately
retirement_warn=(seconds)	Sets when to send retirement warning
retires_on=(date)	Sets retirement date
shutdown_guest	Shuts downs guest operating system of the Service

Method	Use
start	Start the Service
stop	Stop the Service
suspend	Suspend the Service
vms	Show all virtual machines associated with this service
direct_vms	Show virtual machines directly associated with this service
indirect_vms	Show virtual machines associated with lower level services in the hierarchy
root_service	Show the top level service in the hierarchy for the target service
all_service_children	Show all lower level services to the target service in the hierarchy
direct_service_children	Show direct services associated with the target service
indirect_service_children	Show services associated with lower level services of the target service
parent_service	Show the parent service for the target service
description=(new_description)	Sets the service description
remove_from_vmdb	Delete the service from the database
dialog_options	Returns all dialog options
get_dialog_option(key)	Returns a specific dialog option specified by key
set_dialog_option(key, value)	Sets value of a dialog option specified by key

1.33. SERVICE RESOURCE (SERVICE_RESOURCE)

Method	Use
service	Returns the associated service

Method	Use
service_template	Returns the associated service template
resource	Returns the resource for the request
source	Returns the source object

1.34. SERVICE TEMPLATE (SERVICE_TEMPLATE)

Method	Use
group=(group)	Sets group for the service template
owner=(owner)	Sets owner for the service template

1.35. SNAPSHOT (SNAPSHOT)

These methods can be used on Snapshots

Method	Use
vm	Returns Snapshots VM
current?	Checks to see if this is the current snapshot
get_current_snapshot	Returns the current snapshot id
revert_to	Reverts to specified snapshot
remove	Removes specified snapshot

1.36. DATASTORES (STORAGE)

Method	Use
ext_management_systems,	Returns objects Management System
hosts	Returns objects Hosts
vms	Returns objects Virtual Machines
unregistered_vms	Returns objects unregistered Virtual Machines

Method	Use
to_s	Converts object to string
scan	Performs SmartState Analysis on the object

```
  ############################
   #
   # Method: dumpStorage
   # Inputs: $evm.root['storage']
   # Description: Dump Storage information
   #
   ############################
   def dumpStorage(storage)
     $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> Begin
Attributes")
     storage.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix}
- Storage:<#{storage.name}> Attributes - #{k}: #{v.inspect}")}
     $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> End
Attributes")
     $evm.log("info","")

     $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> Begin
Associations")
     storage.associations.sort.each { |assc| $evm.log("info", "#
{@log_prefix} - Storage:<#{storage.name}> Associations - #{assc}")}
     $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> End
Associations")
     $evm.log("info","")

     $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> Begin
Virtual Columns")
     storage.virtual_column_names.sort.each { |vcn| $evm.log("info", "#
{@log_prefix} - Storage:<#{storage.name}> Virtual Columns - #{vcn}: #
{storage.send(vcn)}")}
     $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> End
Virtual Columns")
     $evm.log("info","")
   end
```

1.37. SWITCH (SWITCH)

These methods can be used on Switches.

Method	Use
host	Returns switch's Host
guest_devices	Returns switch's guest devices

Method	Use
lans	Returns switch's lans

1.38. USER (USER)

These methods can be used on the currently logged on user.

Method	Use
current_group	Returns user's assigned internal group
custom_get(key)	Returns the custom key value specified by "key"
custom_keys	Returns an array of custom keys
custom_set(key,value)	Sets custom value for "key" to "value"
email	Returns user's email address
get_ladap_attribute(name)	Returns the value of the specified LDAP attribute
get_ldap_atttribute_names	Returns user's LDAP attribute names
ldap_group	Returns user's assigned LDAP group
miq_requests	Returns user's requests
name	Returns user's name
userid	Returns user's userid
vms	Returns Virtual Machines that this user owns

```
  ############################
  #
  # Method: dumpUser
  # Inputs: $evm.root['user']
  # Description: Dump User information
  #
  ############################
  def dumpUser
    user = $evm.root['user']
    unless user.nil?
      $evm.log("info","#{@method} - User:<#{user.name}> Begin Attributes
[user.attributes]")
      user.attributes.sort.each { |k, v| $evm.log("info", "#{@method} -
User:<#{user.name}> Attributes - #{k}: #{v.inspect}")}
```

```
      $evm.log("info","#{@method} - User:<#{user.name}> End Attributes
[user.attributes]")
      $evm.log("info", "")

      $evm.log("info","#{@method} - User:<#{user.name}> Begin
Associations [user.associations]")
      user.associations.sort.each { |assc| $evm.log("info", "#{@method} -
User:<#{user.name}> Associations - #{assc}")}
      $evm.log("info","#{@method} - User:<#{user.name}> End Associations
[user.associations]")
      $evm.log("info","")

      $evm.log("info","#{@method} - User:<#{user.name}> Begin Virtual
Columns [user.virtual_column_names]")
      user.virtual_column_names.sort.each { |vcn| $evm.log("info", "#
{@method} - User:<#{user.name}> Virtual Columns - #{vcn}: #
{user.send(vcn).inspect}")}
      $evm.log("info","#{@method} - User:<#{user.name}> End Virtual
Columns [user.virtual_column_names]")
      $evm.log("info","")
    end
  end
```

1.39. VIRTUAL MACHINES AND TEMPLATES (VM_OR_TEMPLATE)

The following methods can be used on a virtual machine or template object.

Method	Use
add_disk [a]	Add storage to an existing VM. This method accepts as parameters: disk_name, disk_size_mb, and options to pass values for :thinProvisioned, :dependent, and :persistent. Example: **`vm.add_disk("[#{vm.storage_name}]", size_in_mb, thin_provisioned: true)`**
changed_vm_value?	Checks the 2 most recent drift state captures , and answers whether the specified value changed between them
collect_running_processes	Collects the running processes of the object
create_snapshot(name, desc = nil)	Create a snapshot of the object. The **name** parameter is not used for Red Hat Virtualization virtual machines when creating a snapshot.
custom_get(key)	Get the value of specified Red Hat CloudForms Server key from the object

Method	Use
custom_keys	List all Red Hat CloudForms Server custom keys for the object
custom_set(key, value)	Set a custom Red Hat CloudForms Server key value
datacenter	Returns objects Datacenter
direct_service	Show the direct service relationship of the virtual machine
directories	Returns number of directories on the object
ems_blue_folder (this will be reworked to be more VMware-specific)	Returns objects blue folder from VMware. These are the folders showing in VM and Templates view in VMware
ems_cluster	Returns objects cluster
ems_custom_get(key)	Gets specified key of custom Management System Attribute
ems_custom_keys	List the custom keys defined by the Management System for the object
ems_custom_set(attribute, value)	Sets specified key and value of custom Management System Attribute
ems_folder	Returns objects folder on Management System
ems_ref_string	Returns unique identifier the Management System uses to identify this resource. For example, in VMware a VM would return a value like: "vm-26622"

Method	Use
event_log_threshold? (options)	Searches event log records to determine if an event has occurred x number of times within a defined time frame. Returns true if the number of matching records found are greater or equal to the specified freq_threshold, otherwise it returns false Options values: :message_filter_type - Must be one of "STARTS WITH", "ENDS WITH", "INCLUDES", "REGULAR EXPRESSION" :message_filter_value - <string value to search for> :time_threshold - Options time interval to search. Example: 2.days (Search the past 2 days of event logs) Default 10.days :freq_threshold - Number of occurrences to check for. Default = 2 :source, :event_id, :level, :name - Options filter values
event_threshold?(options)	Checks if an event (or multiple events) have occurred X number of times in N seconds. The values below are used if no data is passed event_threshold?(options = {:time_threshold ⇒ 30.minutes, :event_types ⇒ ["MigrateVM_Task_Complete"], :freq_threshold ⇒ 2})
ext_management_system	Returns objects Management System
files	Returns number of files on the object
get_realtime_metric(metric, range, function)	Returns specified realtime metric
group=(group)	Sets objects group
guest_applications	Returns objects Guest Application list
hardware	Returns objects Hardware
host	Returns objects Host
migrate(host, pool = nil, priority = "defaultPriority", state = nil)	Migrates the object to another host. The only required parameter is host

Method	Use
miq_provision	If VM was created using Red Hat CloudForms Server provisioning, this is the miq_provision task instance that created the VM
operating_system	Returns objects Operating System
owner	Return objects owner
owner=(owner)	Sets objects owner
performances_maintains_value_for_duration?	Based on options given, checks to see if a performance threshold is maintained for a time period Example: vm.performances_maintains_value_for_duration? (:column ⇒ "cpu_usage_rate_average", :operator ⇒ "=", :value ⇒ 3.51, :duration ⇒ 20.minutes)
reboot_guest	Reboots the guest operating system
reconfigured_hardware_value?	Checks if hardware value has been reconfigured
refresh	Refresh power states and relationships of the object
registered?	Is the object registered?
remove_all_snapshots	Remove all of the objects snapshots
remove_from_disk	Removes the object from disk
remove_from_vmdb	Removes the object from the VMDB
remove_snapshot(snapshot_id)	Remove a specific snapshot based on the snapshot_id
resource_pool	Returns objects Resource Pool
retire_now	Retire the object immediately
retirement_warn=(seconds)	Send a retirement warning
retires_on=(date)	Retire the object on date specified
revert_to_snapshot(snapshot_id)	Revert to a snapshot based on the snapshot_id

Method	Use
scan(scan_categories = nil)	Perform SmartState Analysis on the object. Scan_categories is optional
service	Show the top-level service for a virtual machine in a service hierarchy. For the immediate parent service relationship of a virtual machine, use direct_service
shutdown_guest	Shuts down the guest operating system of the object
snapshots	Returns list of snapshots for the object
standby_guest	Puts the operating system on standby
start	Starts the object See Samples/PowerOn_DHOB
stop	Stops the object
storage	Returns objects Datastore
suspend	Suspends the object
to_s	Converts object to string
unlink_storage	Removes the reference to the VM's Datastore
unregister	Unregisters the object from the Management System

[a] Supported for Red Hat Virtualization and VMWare virtual machines

```
  ############################
  #
  # Method: dumpVM
  # Inputs: $evm.root['vm']
  # Description: Dump VM information
  #
  ############################
  def dumpVM(vm)
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Attributes
[vm.attributes]")
    vm.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} -
VM:<#{vm.name}> Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Attributes
[vm.attributes]")
    $evm.log("info","")
```

```
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Associations
[vm.associations]")
    vm.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} -
VM:<#{vm.name}> Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Associations
[vm.associations]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Hardware
Attributes [vm.hardware]")
    vm.hardware.attributes.each { |k,v| $evm.log("info", "#{@log_prefix} -
VM:<#{vm.name}> Hardware - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Hardware
Attributes [vm.hardware]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Hardware
Associations [vm.hardware.associations]")
    vm.hardware.associations.sort.each { |assc| $evm.log("info", "#
{@log_prefix} - VM:<#{vm.name}> hardware Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End hardware
Associations [vm.hardware.associations]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Neworks
[vm.hardware.nics]")
    vm.hardware.nics.each { |nic| nic.attributes.sort.each { |k,v|
$evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> VLAN:<#
{nic.device_name}> - #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Networks
[vm.hardware.nics]")
    $evm.log("info","")

    unless vm.ext_management_system.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin EMS
[vm.ext_management_system]")
      vm.ext_management_system.attributes.sort.each { |ems_k, ems_v|
$evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> EMS:<#
{vm.ext_management_system.name}> #{ems_k} - #{ems_v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End EMS
[vm.ext_management_system]")
      $evm.log("info","")
    end

    unless vm.owner.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Owner
[vm.owner]")
      vm.owner.attributes.each { |k,v| $evm.log("info", "#{@log_prefix} -
VM:<#{vm.name}> Owner - #{k}: #{v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Owner
[vm.owner]")
      $evm.log("info","")
    end

    unless vm.operating_system.nil?
```

```
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Operating
System [vm.operating_system]")
      vm.operating_system.attributes.sort.each { |k, v| $evm.log("info",
"#{@log_prefix} - VM:<#{vm.name}> Operating System - #{k}: #{v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Operating
System [vm.operating_system]")
      $evm.log("info","")
    end

    unless vm.guest_applications.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Guest
Applications")
      vm.guest_applications.each { |guest_app|
guest_app.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} -
VM:<#{vm.name}> Guest Application:<#{guest_app.name}> - #{k}: #
{v.inspect}")}} unless vm.guest_applications.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Guest
Applications")
      $evm.log("info","")
    end

    unless vm.snapshots.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Snapshots")
      vm.snapshots.each { |ss| ss.attributes.sort.each { |k, v|
$evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> Snapshot:<#{ss.name}> -
#{k}: #{v.inspect}")}} unless vm.snapshots.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Snapshots")
      $evm.log("info","")
    end

    unless vm.storage.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin VM Storage
[vm.storage]")
      vm.storage.attributes.sort.each { |stor_k, stor_v| $evm.log("info",
"#{@log_prefix} - VM:<#{vm.name}> Storage:<#{vm.storage.name}> #{stor_k} -
#{stor_v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End VM Storage
[vm.storage]")
      $evm.log("info","")
    end

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Virtual
Columns -----")
    vm.virtual_column_names.sort.each { |vcn| $evm.log("info", "#
{@log_prefix} - VM:<#{vm.name}> Virtual Columns - #{vcn}: #
{vm.send(vcn).inspect}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Virtual Columns
-----")
    $evm.log("info","")
  end
```

```
  ####################
  #
  # Method: createSnapshot
  #
  ####################
```

```
  def createSnapshot(vm, snap_name, snap_desc=snap_name)
    $evm.log("info","#{@method} - VM:<#{vm.name}> Creating Snapshot:<#
{snap_name}> Description:<#{snap_desc}>")
    vm.create_snapshot(snap_name, snap_desc)
  end
```

```
  ####################
  # Red Hat Virtualization Virtual Machines
  # Method: createSnapshot
  #
  ####################

 def createSnapshot(vm, snap_desc=snap_name)
  $evm.log("info","#{@method} - VM:<#{vm.name}> Creating Snapshot:<#
{snap_name}> Description:<#{snap_desc}>")
  vm.create_snapshot(snap_desc)
 end
```

1.39.1. VMs (vm)

Method	Use
add_to_service(service)	Adds the VM to a service object
remove_from_service	Removes the VM from its parent service

The following table lists the metric types available for the **`get_realtime_metric(metric, range, function)`** method for virtual machines.

Metric	Description
v_derived_storage_used	Capacity - Used space in bytes
v_pct_cpu_ready_delta_summation	CPU - Percentage ready
v_pct_cpu_wait_delta_summation	CPU - Percentage wait
v_pct_cpu_used_delta_summation	CPU - Percentage used
v_derived_vm_count	State - Peak average virtual machines (Hourly Count / Daily Average)
v_derived_cpu_reserved_pct	CPU - Percentage available
v_derived_memory_reserved_pct	Memory - Percentage available

The following Ruby snippet demonstrates using the **`get_realtime_metric(metric, range, function)`** method using the **`v_pct_cpu_ready_delta_summation`** metric.

```
host = $evm.root['vm']
cpu_rdy = vm.get_realtime_metric(:v_pct_cpu_ready_delta_summation,
[15.minutes.ago.utc,5.minutes.ago.utc], :avg)
```

1.39.1.1. VMs for Clouds (vm_cloud)

Method	Use
availability_zone	Returns object's availability zone
flavor	Returns object's favor
cloud_network	Returns object's cloud network
cloud_subnet	Returns object's cloud subnet
floating_ip	Returns object's floating IPs
security_groups	Returns object's security groups
key_pairs	Returns object's key pairs

1.40. WINDOWS IMAGE (WINDOWS_IMAGE)

Method	Use
customization_templates	Returns the images customization templates.
pxe_server	Returns the images pxe server.

1.41. CREATING CATEGORIES AND TAGS

These methods are invoked using $evm.execute. See after the table for usage examples.

Method(parameter)	Use
category_exists?	Checks to see if a tag category exists. Usually used with unless or if to then create a category that does not exist. Example: $evm.execute('category_exists?', "department")

Method(parameter)	Use
category_create	Creates a new category, and if only one tag value from this category can be assigned to a configuration item, sets that tag value using the `single_value` option. Example: $evm.execute('category_create', :name ⇒ "department", :single_value ⇒ false, :description ⇒ "Department") **NOTE:** The `category_create` method currently does not work as expected. The user interface shows the value provided for `:description` in the **Display Name** field and the value provided for `:example_text` is displayed in the **Description** field.
tag_exists?	Checks to see if a tag exists in a category. Example: $evm.execute('tag_exists?', "department", "finance")
tag_create	Creates a new tag in the specified category. Example: $evm.execute('tag_create', "department", :name ⇒ "finance", :description ⇒ "Finance")

1.41.1. Category and Tags Methods Example

In this example, the VMDB is checked to see if the **Department** category exists. If it does, then a message is logged. If not, the category is created and a message is logged. Values are then added to the category.

```
if $evm.execute('category_exists?', "department")
  $evm.log("info", "Classification department exists")
else
  $evm.log("info", "Classification department doesn't exist, creating
category")
  $evm.execute('category_create', :name => "department", :single_value =>
false, :description => "Department")
  $evm.log("info", "Adding new tag in Department Category")
  $evm.execute('tag_create', "department", :name => "finance",
:description => "Finance")
```

1.42. QUOTA

Method(parameter)	Use
vms_by_owner, vms_by_group, vms_by_owner_and_group	Collect stats about existing VMs by owner or in the same LDAP group as the owner in the current request. Sample return object: {:class_name⇒"Vm", :count⇒5, :ids⇒[22, 120, 121, 122, 117], :cpu⇒6, :memory⇒5120, :allocated_storage⇒29032972288, :used_storage⇒29032972288}
retired_vms_by_owner, retired_vms_by_group, retired_vms_by_owner_and_group	Collect stats about retired VMs, that have not been deleted from the host, by owner or in the same LDAP group as the owner in the current request. Sample return object: {:class_name⇒"Vm", :count⇒5, :ids⇒[22, 120, 121, 122, 117], :cpu⇒6, :memory⇒5120, :allocated_storage⇒29032972288, :used_storage⇒29032972288}
provisions_by_owner, provisions_by_group	Collect stats based on provisions running on the same day as the current request (based on scheduled date or immediate) by owner or in the same LDAP group as the owner. (Results do not include Provision Requests that have not been approved.) Sample return object: {:class_name⇒"MiqProvisionRequest", :count⇒6, :ids⇒[115, 116, 104, 102, 105, 112], :cpu⇒6, :memory⇒1536, , :storage⇒62914560}
requests_by_owner, requests_by_group	Collect stats based on provision requests made the same day as the request by the same owner or LDAP group as the owner in the current request, regardless of when the provision request is scheduled to run. (Results do not include Provision Requests that have been denied.) Sample return object: {:class_name⇒"MiqProvisionRequest", :count⇒6, :ids⇒[115, 116, 104, 102, 105, 112], :cpu⇒6, :memory⇒1536, , :storage⇒62914560}

Method(parameter)	Use
active_provisions_by_owner, active_provisions_by_group, active_provisions	Collect stats based on currently active provision requests by the same owner or LDAP group as the owner in the current request. Sample return object: {:class_name⇒"MiqProvisionRequest", :count⇒6, :ids⇒[115, 116, 104, 102, 105, 112], :cpu⇒6, :memory⇒1536, , :storage⇒62914560}

APPENDIX A. TERMINOLOGY

Account Role

A designation assigned to a user allowing or restricting a user to parts and functions of the Red Hat CloudForms console.

Action

An execution that is performed after a condition is evaluated.

Alert

Red Hat CloudForms alerts notify administrators and monitoring systems of critical configuration changes and threshold limits in the virtual environment. The notification can take the form of either an email or an SNMP trap.

Analysis Profile

A customized scan of hosts, virtual machines, or instances. You can collect information from categories, files, event logs, and registry entries.

Cloud

A pool of on-demand and highly available computing resources. The usage of these resources are scaled depending on the user requirements and metered for cost.

Red Hat CloudForms appliance

A virtual machine on which the virtual management database (VMDB) and Red Hat CloudForms server reside.

Red Hat CloudForms Console

A web-based interface into the Red Hat CloudForms appliance.

Red Hat CloudForms Role

A designation assigned to a Red Hat CloudForms server that defines what a Red Hat CloudForms server can do.

Red Hat CloudForms Server

The application that runs on the Red Hat CloudForms appliance and communicates with the SmartProxy and the VMDB.

Cluster

Hosts that are grouped together to provide high availability and load balancing.

Condition

A test of criteria triggered by an event.

Discovery

Process run by the Red Hat CloudForms server which finds virtual machine and cloud providers.

Drift

The comparison of a virtual machine, instance, host, cluster to itself at different points in time.

Event

A trigger to check a condition.

Event Monitor

Software on the Red Hat CloudForms appliance which monitors external providers for events and sends them to the Red Hat CloudForms server.

Host

A computer on which virtual machine monitor software is loaded.

Instance/Cloud Instance

A on-demand virtual machine based upon a predefined image and uses a scalable set of hardware resources such as CPU, memory, networking interfaces.

Managed/Registered VM

A virtual machine that is connected to a host and exists in the VMDB. Also, a template that is connected to a provider and exists in the VMDB. Note that templates cannot be connected to a host.

Managed/Unregistered VM

A virtual machine or template that resides on a repository or is no longer connected to a provider or host and exists in the VMDB. A virtual machine that was previously considered registered may become unregistered if the virtual machine was removed from provider inventory.

Provider

A computer on which software is loaded which manages multiple virtual machines that reside on multiple hosts.

Policy

A combination of an event, a condition, and an action used to manage a virtual machine.

Policy Profile

A set of policies.

Refresh

A process run by the Red Hat CloudForms server which checks for relationships of the provider or host to other resources, such as storage locations, repositories, virtual machines, or instances. It also checks the power states of those resources.

Regions

Regions are used to create a central database for reporting and charting. Regions are used primarily to consolidate multiple VMDBs into one master VMDB for reporting.

Resource

A host, provider, instance, virtual machine, repository, or datastore.

Resource Pool

A group of virtual machines across which CPU and memory resources are allocated.

Repository

A place on a datastore resource which contains virtual machines.

SmartProxy

- The SmartProxy is a software agent that acts on behalf of the Red Hat CloudForms appliance to perform actions on hosts, providers, storage and virtual machines.
- The SmartProxy can be configured to reside on the Red Hat CloudForms appliance or on an ESX server version.
- The SmartProxy can be deployed from the Red Hat CloudForms appliance, and provides visibility to the VMFS storage. Each storage location must have a SmartProxy with visibility to it. The SmartProxy acts on behalf of the Red Hat CloudForms appliance.
- If the SmartProxy is not embedded in the Red Hat CloudForms server, it communicates with the Red Hat CloudForms appliance over HTTPS on standard port 443.

SmartState Analysis

Process run by the SmartProxy which collects the details of a virtual machine or instance. Such details include accounts, drivers, network information, hardware, and security patches. This process is also run by the Red Hat CloudForms server on hosts and clusters. The data is stored in

the VMDB.

SmartTags

Descriptors that allow you to create a customized, searchable index for the resources in your clouds and infrastructure.

Storage Location

A device, such as a VMware datastore, where digital information resides that is connected to a resource.

Tags

Descriptive terms defined by a Red Hat CloudForms user or the system used to categorize a resource.

Template

A template is a copy of a preconfigured virtual machine, designed to capture installed software and software configurations, as well as the hardware configuration, of the original virtual machine.

Unmanaged Virtual Machine

Files discovered on a datastore that do not have a virtual machine associated with them in the VMDB. These files may be registered to a provider that the Red Hat CloudForms server does not have configuration information on. Possible causes may be that the provider has not been discovered or that the provider has been discovered, but no security credentials have been provided.

Virtual Machine

A software implementation of a system that functions similar to a physical machine. Virtual machines utilize the hardware infrastructure of a physical host, or a set of physical hosts, to provide a scalable and on-demand method of system provisioning.

Virtual Management Database (VMDB)

Database used by the Red Hat CloudForms appliance to store information about your resources, users, and anything else required to manage your virtual enterprise.

Virtual Thumbnail

An icon divided into smaller areas that summarize the properties of a resource.

Zones

Red Hat CloudForms Infrastructure can be organized into zones to configure failover and to isolate traffic. Zones can be created based on your environment. Zones can be based on geographic location, network location, or function. When first started, new servers are put into the default zone.