Machine Extensibility

vRealize Automation 6.2

vmware®

You can find the most up-to-date technical documentation on the VMware Web site at:

https://docs.vmware.com/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Machine Extensibility

Machine Extensibility provides information about customizing IaaS workflows by using vRealize Orchestrator as well as vRealize Automation Designer and its associated command-line tools.

It covers the following subjects:

- Customizing IaaS machine state change workflows, including calling vRealize Orchestrator workflows from IaaS by using vRealize Orchestrator or vRealize Automation Designer.
- Installing and configuring the vRealize Automation plug-in.
- Installing and configuring vRealize Automation Designer.
- Associating workflows and Distributed Execution Managers by using skills.
- Reference information for the CloudUtil command-line tool.

Intended Audience

This information is intended for workflow developers who want to use vRealize Orchestrator to customize IaaS machine state change workflows.

For information about using Advanced Service Designer to call vRealize Orchestrator workflows, see *Advanced Service Design*.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Updated Information for Machine Extensibility

This *Machine Extensibility* is updated with each release of the product or when necessary.

This table provides the update history of the *Machine Extensibility* .

Revision	Description
EN-001637-01	■ Revised location in "Install vRealize Orchestrator Customization," on page 19. ■ Added `VcoEndpointName` as an additional endpoint option in "Call a vRealize Orchestrator Workflow," on page 33. ■ Added additional endpoints, `GetVcoEndpointByManagementEndpoint` and `GetVcoEndpointByHost`, and added `WorkflowTimeout` as a synchronous activity in "Using vRealize Orchestrator Workflow Activities," on page 32.
EN-001637-00	Initial release.

1 Machine Extensibility Overview

Provisioning or decommissioning a new machine, especially for mission-critical systems, typically requires interacting with a number of different management systems, including DNS servers, load balancers, CMDBs, IP Address Management and other systems.

This chapter includes the following topics:

Machine Life Cycle Extensibility

You can inject custom logic at various predetermined IaaS life cycle stages by leveraging IaaS state change workflows, known as workflow stubs. You can use the workflow stubs to call out to vRealize Orchestrator for bi-directional integration with external management systems.

Creating a state change workflow enables you to trigger the execution of a workflow before the IaaS master workflow enters a specific state. For example, you can create custom workflows to integrate with an external database and record information at different stages of the machine life cycle.

- Create a custom workflow that runs before the master workflow enters the MachineProvisioned state to record such information as machine owner, approvers, and so on.
- Create a custom workflow that runs before a machine enters the MachineDisposing state to record the time at which the machine was destroyed and data such as its resource utilization at last data collection, last logon, and so on.

The master workflow illustrations show the main states of the master workflow, highlighting in yellow the states you can customize by using IaaS workflow stubs. The Customizable State Change Workflows table lists the workflow stubs available, their corresponding place in the master workflow state, and examples of custom logic you could use at each state to extend the machine life cycle.

Figure 1-1. Master workflow states for provisioning machines

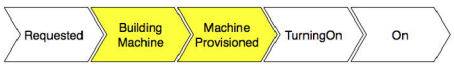


Figure 1-2. Master workflow states for importing machines

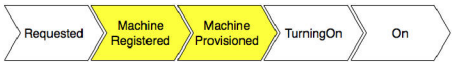


Figure 1-3. Master workflow states for machine lease expiration

Figure 1-4. Master workflow states for disposing a machine

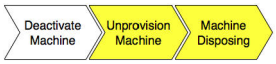


Table 1-1. Customizable State Change Workflows

Master Workflow State	Customizable Workflow Name	Extensibility Examples
BuildingMachine	WFStubBuildingMachine	Prepare for the machine to be created on the hypervisor.Create a configuration management database (CMDB) record, call out to an external system to assign an IP address to a machine, and then during machine disposal, use WFStubMachineDisposing to return the IP address to the pool.
RegisterMachine	WFStubMachineRegistered	Add an imported machine to an application provisioning tool to receive updates and undergo compliance checks.
MachineProvisioned	WFStubMachineProvisioned	The machine exists on the hypervisor, and any additional customizations are completed at this point, for example guest agent customizations. Use this workflow stub to update a configuration management database (CMDB) record with DCHP IP address and storage information. Customizations made by using the WFStubMachineProvisioned are typically reversed by using WFStubUnprovisionMachine.
Expired	WFStubMachineExpired	Move an expired machine to low cost storage to reduce archival costs and update the CMDB record and billing system to reflect storage and cost changes.
UnprovisionMachine	WFStubUnprovisionMachine	Remove machines from active directory accounts. Customizations made by using the WFStubMachineProvisioned are typically reversed by using WFStubUnprovisionMachine.
Disposing	WFStubMachineDisposing	Return IP addresses to the pool.

Choosing a Lifecycle Extensibility Scenario

You can use vRealize Orchestrator or vRealize Automation Designer to extend machine lifecycles.

You can extend machine lifecycles by using vRealize Automation Designer to call out to vRealize Orchestrator, or by using vRealize Orchestrator directly. Both approaches allow you to inject custom logic into predetermined stages of the IaaS machine lifecycle by creating custom vRealize Orchestrator workflows and then insert the custom workflows into the state change workflow stubs. However, you can only restrict custom state change logics to particular blueprints if you are using vRealize Orchestrator directly, and you can only restrict the execution of workflows to specific Distributed Execution Managers by vRealize Automation Designer.

Table 1-2. Choosing a Lifecycle Extensibility Scenario

Scenario	Procedure
Inject custom logic into predetermined stages of the IaaS machine lifecycle and apply that custom logic to specific blueprints.	"Extending Machine Lifecycles by Using vRealize Orchestrator Checklist," on page 15
Inject custom logic into predetermined stages of the IaaS machine lifecycle and apply that custom logic globally to all of your blueprints.	"Extending Machine Lifecycles By Using vRealize Automation Designer Checklist," on page 23
Restrict execution of workflows to specific Distributed Execution Managers by using skills in vRealize Automation Designer. Skills are similar to a tag that you can apply to both workflows and DEM Worker instances. For example, you may want to restrict cloud provisioning workflows to a specific DEM running on a host with the required network access to Amazon URLs.	"Associate Workflows and DEM Workers by Using Skills," on page 41

Extending Machine Lifecycles By Using vRealize Orchestrator 2

You can inject custom logic into predetermined stages of the IaaS machine lifecycle by creating custom vRealize Orchestrator workflows and then using vRealize Orchestrator to insert the custom workflows into the lifecycle of machines built from specific blueprints.

This chapter includes the following topics:

Extending Machine Lifecycles by Using vRealize Orchestrator Checklist

The extending machine lifecycles by using vRealize Orchestrator checklist provides a high-level overview of the steps required to install and configure vRealize Orchestrator to customize IaaS machine lifecycles.

Table 2-1. Extending Machine Lifecycles by Using vRealize Orchestrator Checklist

Task	Details
❑ If you are using a standalone instance of vRealize Orchestrator, install the vRealize Automation plugin. The plugin is already installed in the vRealize Orchestrator embedded in vRealize Automation. You can use the embedded instance for development or proof of concept deployments, but it is not recommended for use in production environments.	"Install the vRealize Automation Plug-In On an External vRealize Orchestrator Server," on page 17
❑ Configure a vRealize Automation host for your vRealize Orchestrator.	"Add a vRealize Automation Host," on page 17
❑ Configure an IaaS host for your vRealize Orchestrator.	"Add an IaaS Host," on page 18
❑ Install the vRealize Orchestrator customizations for extending IaaS machine lifecycles.	"Install vRealize Orchestrator Customization," on page 19
❑ Create a vRealize Automation endpoint for your vRealize Orchestrator instance.	"Create a vRealize Orchestrator Endpoint," on page 26

Table 2-1. Extending Machine Lifecycles by Using vRealize Orchestrator Checklist (Continued)

Task	Details
❑ Use the workflow template provided in the Extensibility subdirectory of the vRealize Automation plugin library to create a custom vRealize Orchestrator workflow to run during the machine lifecycle. You can run multiple workflows in the same state for the same blueprint as long as you nest them under a single wrapper workflow.	For information about developing workflows with vRealize Orchestrator, see the vRealize Orchestrator documentation. For training in vRealize Orchestrator development for vRealize Automation integrations, see the training courses available from VMware Education and the instructional material provided by VMware Learning.
❑ Run the provided workflow that inserts your custom workflow into an IaaS workflow stub and configures a blueprint to call the IaaS workflow stub.	"Assign a State Change Workflow to a Blueprint and Its Virtual Machines," on page 21

Installing the vRealize Automation Plug-In

The vRealize Automation plug-in is installed on the embedded vRealize Orchestrator instance in your vRealize Automation installation. You can use the configuration interface of an external vRealize Orchestrator server to install the vRealize Automation plug-in.

vCenter Orchestrator Integration Prerequisites

If you are using vRealize Automation workflows to execute vRealize Orchestrator workflows that have input or output parameters of type `VC:VirtualMachine`, verify that you have the vRealize Orchestrator workflows for converting virtual machine types between vRealize Orchestrator and IaaS.

The required workflows are included by default in vRealize Orchestrator 5.5 as part of the vCenter plugin.

If you are using vRealize Orchestrator 5.1, install the vRealize Automation integration package for vRealize Orchestrator. Download the package `com.vmware.library.vcenter.vcac–integration.package` from thevRealize Orchestrator community site at http://communities.vmware.com/community/vmtn/server/vcenter/orchestrator. Import the package on each vRealize Orchestrator server that you set up as an endpoint in IaaS.

For information about importing packages to vRealize Orchestrator, refer to the vRealize Orchestrator documentation.

vRealize Automation Plug-In Functional Prerequisites

To install and use the vRealize Automation plug-in, your system must meet certain functional prerequisites.

vRealize Automation

You must have access to a vRealize Automation server. Version 6.2 of the plug-in works with vRealize Automation 6.2.

For information about setting up vRealize Automation, see vRealize Automation *Installation and Configuration*.

vRealize Orchestrator Server

Version 6.2 of the plug-in works with vRealize Orchestrator 6.0.0.

For information about setting up Orchestrator, see *Installing and Configuring VMware vRealize Orchestrator*.

Other Prerequisites

For full plug-in functionality, you must uninstall Web Distributed Authoring and Versioning (WebDAV) from your Microsoft Web Server (IIS) role services.

Install the vRealize Automation Plug-In On an External vRealize Orchestrator Server

You can install the plug-in on an external vRealize Orchestrator server by using the Orchestrator configuration interface.

Prerequisites

- Download the `.vmoapp` file that contains the plug-in.
- Log in to the Orchestrator configuration interface at http://*orchestrator_server*:8283.

Procedure

1 On the **General** tab, click **Install Application**.

2 Upload the vRealize Automation plug-in.

   a Click the magnifying glass icon.

   b Select the `.vmoapp` file to install.

   c Click **Open**.

   d Click **Install**.

   A message appears after successful installation. The vRealize Automation plug-in is installed without a tab in the Orchestrator configuration interface.

3 On the **Startup Options** tab, click **Restart service** to complete the plug-in installation.

What to do next

Configure the vRealize Automation plug-in.

Configuring the vRealize Automation Plug-in for Machine Extensibility

You configure your vRealize Automation and IaaS hosts, install the customizations for machine extensibility, and create a vRealize Automation endpoint for your vRealize Orchestrator instance.

Add a vRealize Automation Host

You can run a workflow to a vRealize Automation host and configure the host connection parameters.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Run** or **Design**.

2 Click the **Workflows** view.

3 Select **Library > vCloud Automation Center > Configuration**.

4 Right-click the **Add a vCAC host** workflow and select **Start workflow**.

5 Enter a unique name for the host in the **Host Name** text box.

6 Enter the URL address of the host in the **Host URL** text box.

   For example: *https://hostname*.

7 (Optional) Enter the name of the tenant in the **Tenant** text box.

To use the full functionality of the plug-in for a tenant, create a dedicated vRealize Automation host for each tenant.

8 Select whether to install the SSL certificates automatically without user confirmation.

9 (Optional) To configure the length of time vRealize Orchestrator waits for a connection or response from vRealize Automation, enter timeout intervals in the **Connection timeout (seconds)** and **Operation timeout (seconds)** text boxes.

10 Select the type of connection to the host from the **Session mode** drop-down menu.

Option	Actions
Shared Session	Enter the credentials for a vRealize Automation user in the **Authentication username** and **Authentication password** text boxes.
Per User Session	Connect by using the credentials of the user that is currently logged in. You must be logged in to the Orchestrator client with the credentials of the vRealize Automation system administrator. To use this option with an external vRealize Orchestrator server, you must run the Register vCO in vCAC Component Registry workflow and restart the Orchestrator server.

11 Click **Submit**.

What to do next

Add a vRealize Automation Infrastructure Administration host.

Add an IaaS Host

You can run a workflow to add the IaaS host of a vRealize Automation host and configure the connection parameters.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Run** or **Design**.

2 Click the **Workflows** view.

3 Select **Library > vCloud Automation Center > Configuration**.

4 Right-click **Add the IaaS host of a vCAC host** and select **Start workflow**.

5 Select the vRealize Automation host for which you want to configure an IaaS host from the **vCAC host** drop-down menu.

6 Click **Next**.

7 Enter a unique name for the host in the **Host Name** text box.

8 Enter the URL of the machine on which your Model Manager is installed.

For example: https://*model_manager_machine.com*.

9 (Optional) To configure the length of time vRealize Orchestrator waits for a connection or response from vRealize Automation, enter timeout intervals in the **Connection timeout (seconds)** and **Operation timeout (seconds)** text boxes.

10 Click **Next**.

11 Select the type of connection to the host from the **Session mode** drop-down menu.

Option	Actions
Shared Session	In the **Authentication username** and **Authentication password** text boxes, enter credentials for a vRealize Automation user who has administrative rights to the machine where the IaaS Model Manager is installed.
Per User Session	Connect by using the credentials of the user that is currently logged in. You must be logged in to the Orchestrator client with the credentials of a vRealize Automation user who has administrative rights to the machine where the IaaS Model Manager is installed.

12 Click **Next**.

13 Enter the name of the Workstation machine and the NetBIOS domain name.

14 Click **Submit**.

Install vRealize Orchestrator Customization

You can run a workflow to install the customized state change workflow stubs and Orchestrator menu operation workflows.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Run** or **Design**.

2 Click the **Workflows** view.

3 Select **Library > vCloud Automation Center > Infrastructure Administration > Extensibility > Installation**.

4 Right-click the **Install vCO customization** workflow and select **Start workflow**.

5 Select an IaaS host.

6 Click **Next**.

7 Choose the lifecycle stages to which you want to add custom logic by selecting one or more state change workflow stubs to install.

8 Click **Submit**.

Create a vRealize Orchestrator Endpoint

vRealize Automation uses vRealize Orchestrator endpoints to run workflows. You can configure multiple endpoints to connect to different vRealize Orchestrator servers. Each endpoint must have a priority.

When executing vRealize Orchestrator workflows, vRealize Automation tries the highest priority vRealize Orchestrator endpoint first. If that endpoint is not reachable, then it proceeds to try the next highest priority endpoint until a vRealize Orchestrator server is available to run the workflow.

Prerequisites

Log in to the vRealize Automation console as an **IaaS administrator**.

Procedure

1 Select **Infrastructure > Endpoints > Endpoints**.

2 Select **New Endpoint > Orchestration > vCenter Orchestrator**.

3 Enter a name and, optionally, a description.

4 Type a URL with the fully qualified name or IP address of the vRealize Orchestrator server and the vRealize Orchestrator port number.

The format depends on the version of the vRealize Orchestrator server.

vRealize Orchestrator version	URL format
5.1	https://*hostname*:*port*
5.5	https://*hostname*:*port*/vco

The transport protocol must be HTTPS. If no port is specified, the default port 8281 is used.

To use the default vRealize Orchestrator instance embedded in the vRealize Appliance, type **`https://vrealize-automation-appliance-hostname:8281/vco`**.

5 Specify the credentials to use to connect to this endpoint.

a Click the ellipsis next to the **Credentials** field.

b Select an existing credential from the list or click **New Credentials** to provide your vRealize Orchestrator credentials.

The credentials you use should have Execute permissions for any vRealize Orchestrator workflows to call from IaaS.

To use the default vRealize Orchestrator instance embedded in the vRealize Appliance, the user name is **`administrator@vsphere.local`** and the password is the administrator password that was specified when configuring SSO.

6 Specify the endpoint priority.

a Click **New Property**.

b Type **`VMware.VCenterOrchestrator.Priority`** in the **Name** text box.

The property name is case sensitive.

c Type an integer greater than or equal to 1 in the **Value** text box.

Lower value means higher priority.

d Click the **Save** icon ( ).

7 Click **OK**.

Customizing IaaS Workflows By Using vRealize Orchestrator

You use a single workflow in vRealize Orchestrator to inject your custom logic into the IaaS workflow stubs and assign your customized life cycles to machine blueprints.

You must design your custom vRealize Orchestrator workflows to accept string inputs. If your custom workflow expects a complex data type, create a wrapper workflow that looks up this complex value and translates it to a string. For an example wrapping workflow, see the sample Workflow template, provided in **Library > vRealize Automation > Infrastructure > Extensibility**.

Assign a State Change Workflow to a Blueprint and Its Virtual Machines

You configure custom vRealize Orchestrator workflows to run at specific stages in the master machine workflow by associating your custom workflow with a state change workflow stub and assigning the workflows to a blueprint.

Prerequisites

Use the workflow template provided in the Extensibility subdirectory of the vRealize Automation plugin library to create a custom workflow to run during the machine lifecycle.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Run** or **Design**.

2 Click the **Workflows** view.

3 Select **Library > vRealize Automation > Infrastructure > Extensibility**.

4 Right-click the **Assign a state change workflow to a blueprint and its virtual machines** workflow and select **Start workflow**.

5 Choose the lifecycle stage at which to run the workflow by selecting a stub from the **vCAC workflow stub to enable** drop-down menu.

6 Select an IaaS host.

7 Click **Next**.

8 Select the blueprint to which you want to assign the workflow.

9 Choose whether or not to apply these workflows to existing machines provisioned by using this blueprint.

10 Select the workflow you want to run during the machine lifecycle.

11 Configure which workflow input values are added as custom properties to the machine.

   a Add vCO workflow inputs as blueprint properties.

   b Add last vCO workflow run input values as blueprint properties.

12 Click **Submit**.

Extending Machine Lifecycles By Using vRealize Automation Designer 3

You can inject custom logic into predetermined stages of the IaaS machine life cycle by using vRealize Automation Designer to directly edit the state change workflow stubs and, optionally, call out to custom vRealize Orchestrator workflows.

This chapter includes the following topics:

Extending Machine Lifecycles By Using vRealize Automation Designer Checklist

The Extending Machine Lifecycles By Using vRealize Automation Designer Checklist provides a high-level overview of the steps required to install and configure vRealize Automation Designer to customize IaaS machine life cycles.

Table 3-1. Extending Machine Lifecycles By Using vRealize Automation Designer Checklist

Task	Details
❑ Download and install vRealize Automation Designer.	"Installing vRealize Automation Designer," on page 24
❑ Create a vRealize Automation endpoint for your vRealize Orchestrator instance.	"Create a vRealize Orchestrator Endpoint," on page 26
❑ Associate your vRealize Orchestrator endpoint with a machine blueprint.	"Associate a vRealize Orchestrator Endpoint with a Blueprint," on page 27
❑ Using vRealize Automation Designer activities, customize an IaaS Workflow stub. Optionally, you can use vRealize Orchestrator workflow activities to call out to custom vRealize Orchestrator workflows.	"Customize an IaaS Workflow," on page 31
❑ After you create a custom state change workflow, a tenant administrator or business group manager must enable it for specific blueprints by adding a custom property.	"Configure a Blueprint to Call a State Change Workflow," on page 36

Installing and Configuring vRealize Automation Designer

You can install vRealize Automation Designer on a Windows machine and configure it to communicate with a remote Model Manager instance. If you are using IaaS workflows to call vRealize Orchestrator workflows, you must also configure the vRealize Orchestrator instance in IaaS.

Installing vRealize Automation Designer

You can install vRealize Automation Designer on a Windows machine and configure it to communicate with a remote Model Manager instance.

vRealize Automation Designer Prerequisites

vRealize Automation Designer is typically installed on a development machine rather than a server.

Supported Operating Systems

- Windows Server 2008 R2 SP1 (64-bit)
- Windows 7 SP1 (32- or 64-bit)
- Windows Vista Business (32- or 64-bit)
- Windows XP SP3 (32- or 64-bit)

System Configuration Requirements

- .NET Framework 4.5 must be installed.
- The vRealize Automation Designer host must have network access to the IaaS Website components (specifically, the Model Manager Web component).
- If the Model Manager is installed remotely, the certificate used for the Model Manager Web component must be trusted on the vRealize Automation Designer host.

Download the vRealize Automation Designer Installer

You can download the vRealize Automation Designer installer from the vRealize Appliance.

Prerequisites

- Log in to the Windows machine as a local administrator.
- If you are using Internet Explorer, verify that Enhanced Security Configuration is not enabled. See `res://iesetup.dll/SoftAdmin.htm`.

Procedure

1. Open a browser.
2. Navigate to the Windows installer download page by using the host name of the (https://*vra-va-hostname.domain.name*:5480/installer/).
3. Click **vRealize Automation Designer**.
4. When prompted, save the installer.

What to do next

"Install vRealize Automation Designer," on page 25.

Install vRealize Automation Designer

The vRealize Automation Designer installer is packaged as Windows installation wizard.

Prerequisites

"Download the vRealize Automation Designer Installer," on page 24.

Procedure

1 Navigate to the directory where you downloaded the installer.

2 Right-click `DesignCenter-Setup.exe` and select **Run as administrator**.

3 On the Welcome page, click **Next**.

4 Read the License Agreement, select **I accept the terms in the License Agreement**, and click **Next**.

5 On the Custom Setup page, click **Next**.

6 Specify the fully qualified domain name and port of the Model Manager Web instance in *hostname:port* format.

The default port is 443.

7 Specify the Model Manager service user credentials.

8 Click **Next**.

The installer validates the combination of Model Manager host and credentials by attempting to access to the Model Manager. If an error is returned, you must provide the correct combination of Model Manager host and credentials before proceeding.

9 Click **Install**.

10 Click **Finish**.

What to do next

You can launch the vRealize Automation Designer from the Windows Start menu by navigating to the installation directory.

Configuring vRealize Orchestrator Endpoints

If you are using vRealize Automation workflows to call vRealize Orchestrator workflows, you must configure the vRealize Orchestrator instance or server as an endpoint.

You can associate a vRealize Orchestrator endpoint with a multi-machine blueprint to make sure that all of the vRealize Orchestrator workflows for machines provisioned from that blueprint are run using that endpoint.

vRealize Automation by default includes an embedded vRealize Orchestrator instance. It is recommended that you use this as your vRealize Orchestrator endpoint for running vRealize Automation workflows in a test environment or creating a proof of concept . For more information about managing the embedded vRealize Orchestrator instance, see *Advanced Service Design*.

You can also install a plug-in on an external vRealize Orchestrator server.

vCenter Orchestrator Integration Prerequisites

If you are using vRealize Automation workflows to execute vRealize Orchestrator workflows that have input or output parameters of type `VC:VirtualMachine`, verify that you have the vRealize Orchestrator workflows for converting virtual machine types between vRealize Orchestrator and IaaS.

The required workflows are included by default in vRealize Orchestrator 5.5 as part of the vCenter plugin.

If you are using vRealize Orchestrator 5.1, install the vRealize Automation integration package for vRealize Orchestrator. Download the package `com.vmware.library.vcenter.vcac-integration.package` from thevRealize Orchestrator community site at http://communities.vmware.com/community/vmtn/server/vcenter/orchestrator. Import the package on each vRealize Orchestrator server that you set up as an endpoint in IaaS.

For information about importing packages to vRealize Orchestrator, refer to the vRealize Orchestrator documentation.

Create a vRealize Orchestrator Endpoint

vRealize Automation uses vRealize Orchestrator endpoints to run workflows. You can configure multiple endpoints to connect to different vRealize Orchestrator servers. Each endpoint must have a priority.

When executing vRealize Orchestrator workflows, vRealize Automation tries the highest priority vRealize Orchestrator endpoint first. If that endpoint is not reachable, then it proceeds to try the next highest priority endpoint until a vRealize Orchestrator server is available to run the workflow.

Prerequisites

Log in to the vRealize Automation console as an **IaaS administrator**.

Procedure

1 Select **Infrastructure > Endpoints > Endpoints**.

2 Select **New Endpoint > Orchestration > vCenter Orchestrator**.

3 Enter a name and, optionally, a description.

4 Type a URL with the fully qualified name or IP address of the vRealize Orchestrator server and the vRealize Orchestrator port number.

The format depends on the version of the vRealize Orchestrator server.

vRealize Orchestrator version	URL format
5.1	https://*hostname*:*port*
5.5	https://*hostname*:*port*/vco

The transport protocol must be HTTPS. If no port is specified, the default port 8281 is used.

To use the default vRealize Orchestrator instance embedded in the vRealize Appliance, type **`https://vrealize-automation-appliance-hostname:8281/vco`**.

5 Specify the credentials to use to connect to this endpoint.

a Click the ellipsis next to the **Credentials** field.

b Select an existing credential from the list or click **New Credentials** to provide your vRealize Orchestrator credentials.

The credentials you use should have Execute permissions for any vRealize Orchestrator workflows to call from IaaS.

To use the default vRealize Orchestrator instance embedded in the vRealize Appliance, the user name is **`administrator@vsphere.local`** and the password is the administrator password that was specified when configuring SSO.

6 Specify the endpoint priority.

a Click **New Property**.

b Type **`VMware.VCenterOrchestrator.Priority`** in the **Name** text box.

The property name is case sensitive.

c Type an integer greater than or equal to 1 in the **Value** text box.

Lower value means higher priority.

d Click the **Save** icon ( ).

7 Click **OK**.

Associate a vRealize Orchestrator Endpoint with a Blueprint

You can specify a particular vRealize Orchestrator endpoint to use with a blueprint.

When IaaS runs a vRealize Orchestrator workflow for any machine provisioned from this blueprint, it always uses the associated endpoint. If the endpoint is not reachable, the workflow fails.

Prerequisites

Log in to the vRealize Automation console as a **tenant administrator** or **business group manager**.

Procedure

1 Select **Infrastructure > Blueprints > Blueprints**.

2 Create a new blueprint or edit an existing blueprint.

If you are editing an existing blueprint, the vRealize Orchestrator endpoint you specify only applies to new machines provisioned from the updated blueprint. Existing machines provisioned from the blueprint continue to use the highest priority endpoint unless you manually add this property to the machine.

3 Click the **Properties** tab.

a Click **New Property**.

b Type `VMware.VCenterOrchestrator.EndpointName` in the **Name** text box.

The property name is case sensitive.

c Type the name of a vRealize Orchestrator endpoint in the **Value** text box.

d Click the **Save** icon ( ).

4 Click **OK**.

Customizing IaaS Workflows By Using vRealize Automation Designer

VMware provides a number of workflows that you can customize using the vRealize Automation Designer. These include state change workflows and menu operation workflows.

IaaS workflows are created using Microsoft Windows Workflow Foundation 4, part of .NET Framework 4. For information on Windows Workflow Foundation and workflow creation, refer to the Microsoft documentation. vRealize Automation also provides several vRealize Automation Designer activities for running and monitoring vRealize Orchestrator workflows.

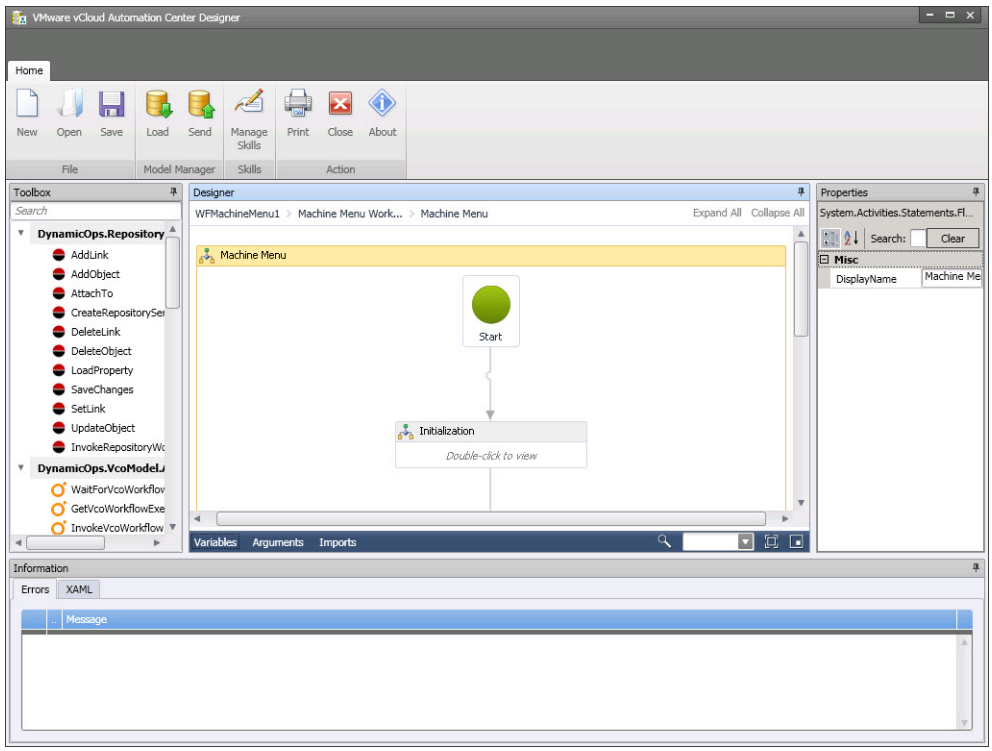
The customizable workflow templates provided by VMware demonstrate best practices for structuring workflows with separate sequences for initialization, custom logic, and finalization. The entire workflow is wrapped in a TryCatch block for error handling. Any uncaught or rethrown exceptions are logged by the Distributed Execution Manager that executes the workflow.

After you create a custom IaaS workflow, a blueprint author must enable the workflow on specific blueprints.

The vRealize Automation Designer Console

The vRealize Automation Designer console provides a visual workflow editor for customizing IaaS workflows.

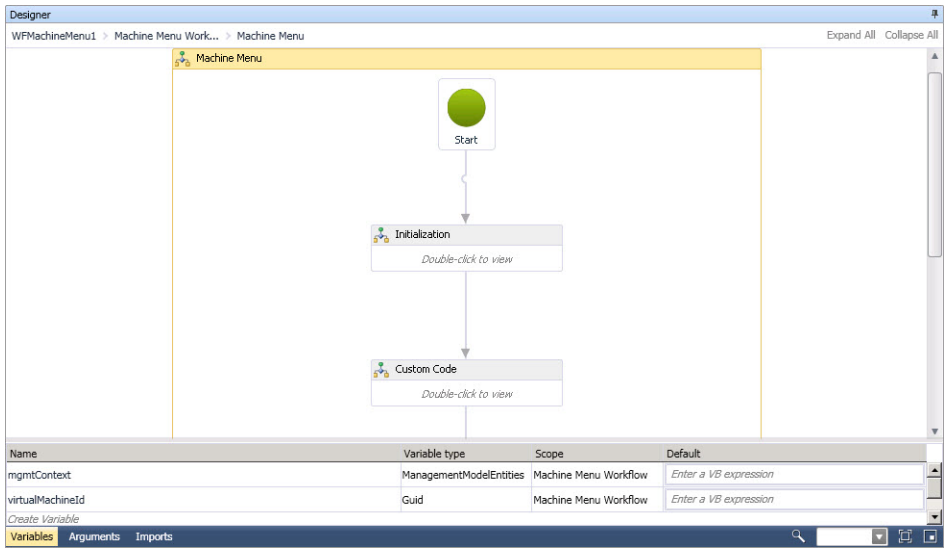
You must have local administrator rights on the vRealize Automation Designer host (typically a development machine) in order to launch the vRealize Automation Designer console.



The Toolbox pane on the left provides access to the vRealize Automation workflow activity library. You can drag activities from the toolbox onto the Designer pane to add them to a workflow. The Properties pane displays the configurable properties of the currently selected activity on the Designer pane. This interface is very similar to the workflow designer in Visual Studio.

The detail tabs at the bottom of the Designer pane enable you to display and edit variables within the scope of the selected activity or arguments to the selected activity.

NOTE Variables and arguments are both specified as Visual Basic expressions. However, variable names are not case sensitive while argument names are case sensitive. For information about valid arguments for the IaaS workflow activities, see Chapter 6, "vRealize Automation Workflow Activity Reference," on page 57.



The Imports tab displays imported namespaces from which you can select entity types to add to the workflow.

The collapsible Information pane at the bottom of the console displays any errors in configuring activities and provides access to the XAML representation of the workflow.

IaaS Workflow Types

You can customize two types of workflows by using vRealize Automation Designer: state change workflows and menu operation workflows.

- A state change workflow is executed when the master workflow transitions between states, for example at a particular stage during the process of provisioning a new machine.
- A menu operation workflow is executed when a user selects an option from the Action menu in the service catalog or from the machine menu in the Infrastructure tab.

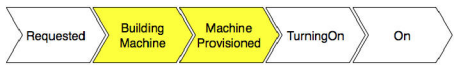
State Change Workflows

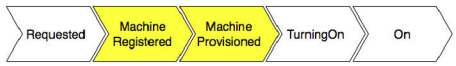
Creating a state change workflow enables you to trigger the execution of a workflow before the IaaS master workflow enters a specific state.

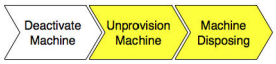
For example, you can create custom workflows to integrate with an external database and record information at different stages of the machine life cycle:

- Create a custom workflow that runs before the master workflow enters the MachineProvisioned state to record such information as machine owner, approvers and so on.
- Create a custom workflow that runs before a machine enters the MachineDisposing state to record the time at which the machine was destroyed and data such as its resource utilization at last data collection, last logon, and so on.

The following illustrations show the main states of the master workflow.







vRealize Automation Designer provides a customizable workflow for each of these states.

Table 3-2. Customizable State Change Workflows

Master Workflow State	Customizable Workflow Name
BuildingMachine	WFStubBuildingMachine
Disposing	WFStubMachineDisposing
Expired	WFStubMachineExpired
MachineProvisioned	WFStubMachineProvisioned
RegisterMachine	WFStubMachineRegistered
UnprovisionMachine	WFStubUnprovisionMachine

Configuring a State Change Workflow Overview

You can customize a state change workflow by using vRealize Automation Designer. A blueprint author can then enable it for specific blueprints.

The following is a high-level overview of the steps required to enable state change workflows:

1 A workflow developer customizes one of the state change workflow templates by using vRealize Automation Designer. See "Customize an IaaS Workflow," on page 31.

  Any IaaS workflow can call a vRealize Orchestrator workflow. For more information, see "Using vRealize Orchestrator Workflow Activities," on page 32.

2 A tenant administrator or business group manager configures a blueprint to call the customized workflow for machines provisioned from that blueprint. See "Configure a Blueprint to Call a State Change Workflow," on page 36.

Menu Operation Workflows

A menu operation workflow is executed when a user selects an option from the Actions menu in the service catalog or the machine menu in the Infrastructure tab.

For example, you can create a custom workflow that enables a user to create a support ticket related to a machine by selecting Raise Support Issue from the machine menu.

vRealize Automation Designer provides templates for customizing menu operation workflows.

In addition to the workflow definition, a menu operation workflow depends on an operation configuration file, which defines the aspects of the custom menu option such as the display text, which roles have access to it, and the machine states for which the operation is available.

Note A service architect can define custom actions for any catalog item by using the Advanced Service Designer. For more information, see *Advanced Service Design*. Creating custom actions for IaaS machines other than those provisioned by using vSphere or vCloud Director requires vRealize Automation 6.1 or later.

Configuring a Menu Operation Workflow Overview

You can customize a menu operation workflow by using vRealize Automation Designer and the CloudUtil command-line utility. A blueprint author can then enable it for specific blueprints.

The following is a high-level overview of the steps required to enable menu operation workflows:

1 A workflow developer customizes one of the menu operation workflow templates by using vRealize Automation Designer. See "Customize an IaaS Workflow," on page 31.

Any IaaS workflow can call a vRealize Orchestrator workflow. For more information, see "Using vRealize Orchestrator Workflow Activities," on page 32.

2 A workflow developer configures the menu operation in the Model Manager. See "Configure a Menu Operation," on page 37.

3 A workflow developer registers the new menu operation with the service catalog. See "Register New Menu Operations with the Service Catalog," on page 39.

4 A tenant administrator or business group manager configures a blueprint to enable the menu operation for machines provisioned from that blueprint. See "Configure a Blueprint to Enable a Menu Operation Workflow," on page 40.

If the menu operation is intended to be used in the service catalog, it must also be entitled to users. For more information, see *Tenant Administration*.

Customize an IaaS Workflow

vRealize Automation Designer enables you to edit the customizable workflows and update workflows in the Model Manager.

Prerequisites

Launch the vRealize Automation Designer.

Procedure

1 Click **Load**.

2 Select the workflow that you want to customize.

Option	Description
WFMachineMenu*N*	Customizable menu operation workflow
WFStubBuildingMachine	Customizable state change workflow that executes before a machine enters the BuildingMachine state
WFStubMachineDisposing	Customizable state change workflow that executes before a machine enters the Disposing state
WFStubMachineExpired	Customizable state change workflow that executes before a machine enters the Expired state
WFStubMachineProvisioned	Customizable state change workflow that executes before a machine enters the MachineProvisioned state
WFStubMachineRegistered	Customizable state change workflow that executes before a machine enters the RegisterMachine state
WFStubUnprovisionMachine	Customizable state change workflow that executes before a machine enters the UnprovisionMachine state

3 Click **OK**.

The workflow displays in the Designer pane.

4 Customize the workflow by dragging activities from the Toolbox to the Designer pane and configuring their arguments.

5 When you are finished editing the workflow, update the workflow in the Model Manager by clicking **Send**.

The workflow is saved and appears as a new revision in the list the next time you load a workflow. You can access an earlier version of a workflow at any time. See "Revert to a Previous Revision of a Workflow," on page 40.

Using vRealize Orchestrator Workflow Activities

You can use vRealize Automation Designer activities to call vRealize Orchestrator workflows either synchronously or asynchronously.

A vRealize Orchestrator endpoint is specified in one of the following ways:

- `VirtualMachineId` is the name of the variable representing the virtual machine ID. A virtual machine with this ID is selected and the value that is retrieved from the `VMware.VCenterOrchestrator.EndpointName` custom property for a virtual machine is used as the vRealize Orchestrator endpoint name.
- `GetVcoEndpointByManagementEndpoint` returns the value of a custom property on a specified `ManagementEndpoint` object. If the `CustomPropertyName` is not specified, the value of the `VMware.VCenterOrchestrator.EndpointName` property is used.
- `GetVcoEndpointByHost` returns the value of a custom property on a specified host. If the `CustomPropertyName` is not specified, the value of the `VMware.VCenterOrchestrator.EndpointName` property is used.

Synchronous Execution

The `InvokeVcoWorkflow` activity calls a vRealize Orchestrator workflow and blocks further execution of its parent IaaS workflow until the vRealize Orchestrator workflow completes. The activity returns the output parameters for the vRealize Orchestrator workflow.

In addition, the synchronous execution supports the following property:

- `WorkflowTimeout` is a timeout value in seconds. If the vRealize Orchestrator workflow does not finish in the specified time, an exception is generated rather than blocking the workflow until a response is returned. If no value is defined or a value of zero is supplied, the timeout is not activated. The workflow status is checked every 10 seconds during that period unless the polling time is modified for the endpoint by specifying a value in the `VMware.VCenterOrchestrator.PollingInterval` custom property.

Asynchronous Execution

The `InvokeVcoWorkflowAsync` activity is a "fire and forget" activity that calls a vRealize Orchestrator workflow and continues to execute activities in the IaaS workflow without waiting for the vRealize Orchestrator workflow to complete.

The activity returns either a unique workflow execution token that can be used to monitor the workflow or an error if the REST API call to the vRealize Orchestrator server failed (for example, if the server could not be reached).

Two additional activities are available for use with this activity:

- `GetVcoWorkflowExecutionStatus` enables you to poll the vRealize Orchestrator workflow for its status.
- `WaitForVcoWorkflowCompletion` enables you to block further execution of the IaaS workflow until the vRealize Orchestrator workflow has completed or timed out. You can use this activity to retrieve the results of a vRealize Orchestrator workflow that you execute asynchronously.

Call a vRealize Orchestrator Workflow

You can use either the `InvokeVcoWorkflow` or the `InvokeVcoWorkflowAsync` activity to call a vRealize Orchestrator workflow from an IaaS workflow.

Some vRealize Orchestrator workflows require user interaction during execution. For these workflows, the user prompt appears in the vRealize Orchestrator client rather than in the vRealize Automation console, so it is not apparent to the end user in vRealize Automation that a workflow is waiting for input.

To avoid workflows that block on user input, do not call vRealize Orchestrator workflows that require user interaction from IaaS workflows.

Procedure

1 In vRealize Automation Designer, open a workflow and navigate to the context where you want to call a vRealize Orchestrator workflow.

2 Drag the `InvokeVcoWorkflow` or the `InvokeVcoWorkflowAsync` activity into the Designer pane.

3 Select the vCenter Orchestrator workflow to execute.

   a Under General, click the ellipsis next to Workflow.

   b In the Browse for vCO workflow dialog box, select a workflow.

   c Click **OK**.

   The Inputs and Outputs sections display the input and output parameters of the selected workflow.

4 In the Properties pane, specify one of the following target parameters.

   - `VirtualMachineId` is the name of the variable representing the virtual machine ID. A virtual machine with this ID is selected and the value that is retrieved from the `VMware.VCenterOrchestrator.EndpointName` custom property for a virtual machine is used as the vRealize Orchestrator endpoint name.
   - `VcoEndpointName` is the endpoint name that is used to run the workflow. If specified, this value overrides the `VirtualMachineId` value when selecting the vRealize Orchestrator endpoint.
   - `WorkflowTimeout` is a timeout value in seconds. If the vRealize Orchestrator workflow does not finish in the specified time, an exception is generated rather than blocking the workflow until a response is returned. If no value is defined or a value of zero is supplied, the timeout is not activated. The workflow status is checked every 10 seconds during that period unless the polling time is modified for the endpoint by specifying a value in the `VMware.VCenterOrchestrator.PollingInterval` custom property.

5 Specify the parameters for the vRealize Orchestrator workflow.

   - Enter the values in the activity in the Designer pane.
   - In the Properties pane, click the ellipsis next to **InputParameters** or **OutputParameters** to open the Parameters dialog box. This dialog box displays the IaaS type of each parameter. If the parameter type appears in bold, the parameter is required.

   Point to the text box for any parameter to view a tooltip indicating the vRealize Orchestrator type.

   If you are using the `InvokeVcoWorkflowAsync` activity, the output parameters of the vRealize Orchestrator workflow are displayed with their corresponding types for informational purposes, but you cannot specify an expression for the parameter in this activity.

What to do next

To retrieve the results of a workflow that you execute asynchronously, use the `WaitForVcoWorkflowCompletion` activity.

Get the Status of a vRealize Orchestrator Workflow

You can check the status of a vRealize Orchestrator workflow that was called with the `InvokeVcoWorkflowAsync` activity using the `GetVcoWorkflowExecutionStatus` activity.

Prerequisites

"Call a vRealize Orchestrator Workflow," on page 33 using the `InvokeVcoWorkflowAsync` activity.

Procedure

1 In vRealize Automation Designer, open a workflow where you have used the `InvokeVcoWorkflowAsync` activity.

2 Navigate to the context where you want to check the status of the vRealize Orchestrator workflow.

3 Drag the `GetVcoWorkflowExecutionStatus` activity into the Designer pane.

4 In the Properties pane, specify the name of the variable representing the virtual machine ID in `VirtualMachineId.`

The customizable workflows contain a variable by default named `virtualMachineId` that is set during initialization.

5 Create a variable of type `DynamicOps.VcoModel.Common.VcoWorkflowExecutionToken`.

6 Specify the name of the token variable as the `executionToken` output parameter on the `InvokeVcoWorkflowAsync` activity.

7 Specify the same variable name as the `WorkflowExecutionToken` property of the `GetVcoWorkflowExecutionStatus` activity.

8 Create a variable of type string.

9 Specify the name of the string variable as the `VcoWorkflowExecutionStatus` property of the `GetVcoWorkflowExecutionStatus` activity.

When the workflow runs, the value of the `VcoWorkflowExecutionStatus` variable is set to the status of the vRealize Orchestrator workflow.

Get the Results of a vRealize Orchestrator Workflow

If you want to call a vRealize Orchestrator workflow asynchronously and then retrieve the results of the completed workflow at a later point, you can use the `WaitForVcoWorkflowCompletion` activity.

The `WaitForVcoWorkflowCompletion` activity blocks execution of the IaaS workflow until the vRealize Orchestrator workflow has completed or a timeout is reached. The activity returns the results of the vRealize Orchestrator workflow if it completes successfully, an error if the workflow fails, or null if the workflow times out.

Prerequisites

"Call a vRealize Orchestrator Workflow," on page 33 using the `InvokeVcoWorkflowAsync` activity.

Procedure

1 In vRealize Automation Designer, open a workflow where you have used the `InvokeVcoWorkflowAsync` activity.

2 Navigate to the context where you want to retrieve the results of the vRealize Orchestrator workflow.

3 Drag the `WaitForVcoWorkflowCompletion` activity into the Designer pane.

4 In the Properties pane, specify the name of the variable representing the virtual machine ID in `VirtualMachineId.`

   The customizable workflows contain a variable by default named `virtualMachineId` that is set during initialization.

5 Create a variable of type DynamicOps.VcoModel.Common.VcoWorkflowExecutionToken.

6 Create a variable of type `DynamicOps.VcoModel.Common.VcoWorkflowExecutionToken`.

7 Specify the name of the token variable as the `executionToken` output parameter on the `InvokeVcoWorkflowAsync` activity.

8 Specify the same variable name as the `WorkflowExecutionToken` property of the `WaitForVcoWorkflowCompletion` activity.

9 Retrieve the output of the vRealize Orchestrator workflow.

   a Create a variable of type `DynamicOps.VcoModel.Common.VcoWorkflowExecutionResult`.

   b Specify the name of the results variable as the `WorkflowOutput` property of the `WaitForVcoWorkflowCompletion` activity.

      When the workflow runs, the value of the variable is set to the results of the vRealize Orchestrator workflow, if any.

vRealize Orchestrator and IaaS Object Types

When you use either the `InvokeVcoWorkflow` or the `InvokeVcoWorkflowAsync` activity in vRealize Automation Designer, input and output properties for the activity are automatically populated based on the parameters of the vRealize Orchestrator workflow that you select.

vRealize Orchestrator primitive types are converted into the following IaaS types:

Table 3-3. vRealize Orchestrator and IaaS Object Types

vRealize Orchestrator Type	IaaS Type
string	string
boolean	bool
number	decimal
SecureString	string
Text	string
Array/T	Array<T>
Properties	Dictionary<string,object>
Date	DateTime
VC:VirtualMachine	VirtualMachine

Note If you are using vRealize Orchestrator 5.1, you must have installed the vRealize Automation integration package to enable the conversion of `VC:VirtualMachine` object types to `VirtualMachine`.

All other vRealize Orchestrator types are converted to the IaaS type `VcoSdkObject`.

Configure a Blueprint to Call a State Change Workflow

After you create a custom state change workflow, a tenant administrator or business group manager must enable it for specific blueprints by adding a custom property.

Each state change workflow is associated with a specific custom property. When a machine is entering a state with a corresponding state change workflow, IaaS checks to see if the machine has the corresponding custom property; if so, the associated workflow is executed. For example, if a machine has the custom property `ExternalWFStubs.MachineProvisioned`, the `WFStubMachineProvisioned` workflow is executed before the master workflow enters the MachineProvisioned state.

While custom properties can be applied to a machine from a number of sources, typically the property for a state change workflow is specified in a blueprint, enabling the workflow for all machines provisioned from that blueprint.

Prerequisites

Log in to the vRealize Automation console as a **tenant administrator** or **business group manager**.

Procedure

1 Select **Infrastructure > Blueprints > Blueprints**.

2 Point to the name of a blueprint and click **Edit**.

3 Click the **Properties** tab.

4 Click **New Property**.

5 Type the name of the custom property associated with the workflow you want to enable in the **Name** text box.

Customizable Workflow Name	Associated Property Name
WFStubMachineProvisioned	ExternalWFStubs.MachineProvisioned
WFStubBuildingMachine	ExternalWFStubs.BuildingMachine
WFStubMachineDisposing	ExternalWFStubs.MachineDisposing
WFStubUnprovisionMachine	ExternalWFStubs.UnprovisionMachine
WFStubMachineRegistered	ExternalWFStubs.MachineRegistered
WFStubMachineExpired	ExternalWFStubs.MachineExpired

6 Leave the **Value** text box blank.

The workflow depends on the presence of the property, not on any particular value.

7 Click the **Save** icon ( ).

8 Click **OK**.

The workflow is now enabled for new machines that are provisioned from this blueprint.

Configuring a Menu Operation Workflow

After you customize a menu operation workflow, additional configuration is required before it is available to users in the vRealize Automation console.

Configure a Menu Operation

To configure a menu operation, you create an operation configuration file and install it in the Model Manager.

Procedure

1 Create an Operation Configuration File on page 37

The operation configuration file is required for menu operation workflows. It specifies the aspects of the custom menu option in the vRealize Automation console such as the display text, which roles have access to the option, and the machine states for which the option is available.

2 Install an Operation in the Model Manager on page 39

You install an operation in the Model Manager by using the CloudUtil command-line utility.

What to do next

If the menu operation is intended to be used in the service catalog, it must be registered with the service catalog so that it can be entitled to users. "Register New Menu Operations with the Service Catalog," on page 39.

Create an Operation Configuration File

The operation configuration file is required for menu operation workflows. It specifies the aspects of the custom menu option in the vRealize Automation console such as the display text, which roles have access to the option, and the machine states for which the option is available.

Procedure

1 Create a new XML file.

```
<?xml version="1.0" encoding="utf-8"?>
```

2 Create the root element `customOperations`.

```
<customOperations xmlns="http://www.dynamicops.com/schemas/2009/OperationConfig/">
</customOperations>
```

The element must specify the XML namespace `http://www.dynamicops.com/schemas/2009/OperationConfig/`.

3 For each operation you want to define, add an `operation` element within `customOperations`.

```
<operation name="WFMachineMenu1" displayName="Execute Machine Menu task">
</operation>
```

The `operation` element takes the following attributes:

Attribute	Description
name	The name of the workflow that this operation executes.
displayName	A descriptive label for the option in the machine menu.

4 Specify the roles to grant access to the menu operation.

a Add the `authorizedTasks` element.

```
<operation name="WFMachineMenu1" displayName="Execute Machine Menu task">
  <authorizedTasks>
  </authorizedTasks>
</operation>
```

b For each role that you want to grant access to the operation, add a `task` element, for example:

```
<authorizedTasks>
  <task>VRM User Custom Event</task>
  <task>VRM Support Custom Event</task>
  <task>Group Administrator Custom Event</task>
  <task>Enterprise Administrator Custom Event</task>
  <task>VRM Administrator Custom Event</task>
</authorizedTasks>
```

The valid contents of the `task` element are as follows:

Element content	Description
VRM User Custom Event	Grants access to the operation for all users.
VRM Support Custom Event	Grants access to the operation for support users.
Group Administrator Custom Event	Grants access to the operation for business group managers.
Enterprise Administrator Custom Event	Grants access to the operation for fabric administrators.
VRM Administrator Custom Event	Grants access to the operation for IaaS administrators only.

5 (Optional) Specify the machine states for which the operation is available.

a Add the `machineStates` element.

```
<operation name="WFMachineMenu1" displayName="Execute Machine Menu task">
  <machineStates>
  </machineStates>
</operation>
```

b For each state in which the operation should be available, add a `state` element.

```
<machineStates>
  <state>On</state>
  <state>Off</state>
</machineStates>
```

The value may be any of the possible machine states. For a full list of machine states, see *IaaS Configuration for Virtual Platforms*, *IaaS Configuration for Physical Machines*, or *IaaS Configuration for Cloud Platforms*.

If the element is omitted, the operation is available for all machine states.

The following is an example of a complete operation configuration file:

```
<?xml version="1.0" encoding="utf-8" ?>
<customOperations xmlns="http://www.dynamicops.com/schemas/2009/OperationConfig/">
  <operation name="WFMachineMenu1" displayName="Execute Machine Menu task">
    <authorizedTasks>
      <task>VRM User Custom Event</task>
      <task>VRM Support Custom Event</task>
      <task>Group Administrator Custom Event</task>
```

```
      <task>Enterprise Administrator Custom Event</task>
      <task>VRM Administrator Custom Event</task>
    </authorizedTasks>
    <machineStates>
      <state>On</state>
      <state>Off</state>
    </machineStates>
  </operation>
</customOperations>
```

Install an Operation in the Model Manager

You install an operation in the Model Manager by using the CloudUtil command-line utility.

Prerequisites

"Create an Operation Configuration File," on page 37.

Procedure

1 Open an elevated command prompt.

2 Run the CloudUtil.exe command with the following arguments.

- `CloudUtil.exe Operation-Create -c <path to operation definition file>`
- Optionally, you can specify a Model Manager host and request a stack trace in case of error.

  ```
  CloudUtil.exe Operation-Create -c <path to operation definition file>
  --repository <Model Manager Root URI> -v
  ```

What to do next

If the menu operation is intended to be used in the service catalog, it must be registered with the service catalog so that it can be entitled to users. "Register New Menu Operations with the Service Catalog," on page 39.

Register New Menu Operations with the Service Catalog

After installing new menu operations, the workflow developer must register them with the service catalog so they can be entitled to users.

Prerequisites

- "Configure a Menu Operation," on page 37.
- Log in to the IaaS Model Manager host as a local user with **administrator** privileges.

Procedure

1 Open an elevated command prompt.

2 Navigate to the IaaS root installation directory.

In a typical installation, this is `C:\Program Files (x86)\VMware\vCAC`.

3 Navigate to `Server\Model Manager Data\Cafe`.

4 Execute the following command:

```
Vcac-Config.exe RegisterCatalogTypes -v
```

What to do next

A tenant administrator or business group manager must entitle the new action before it is available to users in the service catalog. For more information, see *Tenant Administration*.

Configure a Blueprint to Enable a Menu Operation Workflow

You enable a menu operation workflow for machines provisioned from a specific blueprint by updating the security configuration for the blueprint.

Prerequisites

Log in to the vRealize Automation console as a **tenant administrator** or **business group manager**.

Procedure

1 Select **Infrastructure > Blueprints > Blueprints**.

2 Point to the name of a blueprint and click **Edit**.

3 Click the **Actions** tab.

4 Select the checkbox that corresponds to the operation that you want to enable.

5 Click **OK**.

The menu operation is now enabled for machines provisioned from this blueprint and available to all user roles specified in the operation configuration file.

What to do next

If the menu operation is intended to be used in the service catalog, it must also be entitled to users. For more information, see *Tenant Administration*.

Revert to a Previous Revision of a Workflow

The Load Workflow dialog displays all revisions of a workflow in the Model Manager so that you have access to the full version history of the workflows.

Each time you send a workflow to the Model Manager, the Revision and Time Stamp are updated.

Prerequisites

Launch the vRealize Automation Designer console.

Procedure

1 Click **Load**.

2 Select the revision of the workflow that you want to revert to.

The original workflows provided by VMware are revision 0 (zero).

3 Click **OK**.

4 Update the workflow in the Model Manager by clicking **Send**.

The earlier revision becomes the latest revision in the Model Manager. For example, if you have created revisions 1 and 2 of a workflow, then load and save revision 0, revisions 0 and 3 are now identical and you have returned the workflow to the version provided by VMware.

Workflows and Distributed Execution Management

4

You can use skills to restrict execution of workflows to specific Distributed Execution Managers.

A skill is similar to a tag that you can apply to both workflows and DEM Worker instances. If a workflow is not associated with any skills, any DEM Worker can execute it. If a workflow is associated with one or more skills, then only DEM Workers that are associated with all of the same skills can execute it.

Skills are useful when a particular workflow requires a DEM installed on a host with specific prerequisites. For example, you may want to restrict cloud provisioning workflows to a specific DEM running on a host with the required network access to Amazon URLs.

Skills can also be used to associate workflows with a particular data center location. For example, you might install one DEM in your Boston data center and another in your London data center, and use skills to direct certain operations to one data center or the other.

This chapter includes the following topics:

Associate Workflows and DEM Workers by Using Skills

You associate workflows with a specific DEM Worker or set of Worker instances by adding a skill to the Model Manager and then associating the skill with one or more workflows and DEM Workers.

Prerequisites

Launch the vRealize Automation Designer console.

Procedure

1 On the ribbon, click **Manage Skills**.

2 In the text field at the upper left of the Manage Skills dialog, type the name of a new skill and click the Add button.

The skill name must be unique. If the name of the new skill matches the name of an existing skill, the Add button is unavailable.

3 Select the name of the skill in the list on the left.

4 Associate the skill with one or more DEM Workers.

   a Click the **Add** icon () next to Distributed Execution Managers.

   b In the Select DEMs dialog, select one or more DEM Worker instances.

   c Click **OK**.

5 Associate the skill with one or more Workflows.

   a Click the **Add** icon () next to Workflows.

   b In the Select Workflows dialog, select one or more workflows.

   c Click **OK**.

   The workflows associated with this skill can only be executed by the DEM Workers that are associated with this skill.

6 When you are done adding skills and associating them with DEM workers and workflows, click **OK** to close the Manage Skills dialog and save your changes to the Model Manager.

Remove Associations between Skills and DEM Workers

When you remove the association between a skill and a DEM Worker, that Worker instance can no longer execute the workflows associated with the skill.

Prerequisites

Launch the vRealize Automation Designer console.

Procedure

1 On the ribbon, click **Manage Skills**.

2 In the Manage Skills dialog, select the name of the skill in the list on the left.

3 Select the name of one or more DEM Worker instances from the Distributed Execution Managers list and click the **Remove** icon ().

4 Click **OK** to close the Manage Skills dialog and save your changes to the Model Manager.

Remove Associations between Skills and Workflows

When you remove the association between a skill and a workflow, that workflow is no longer restricted to the DEM Workers that are associated with the same skill.

Prerequisites

Launch the vRealize Automation Designer console.

Procedure

1 On the ribbon, click **Manage Skills**.

2 In the Manage Skills dialog, select the name of the skill in the list on the left.

3 Select the name of one or more workflows from the Workflows list and click the **Remove** icon ().

4 Click **OK** to close the Manage Skills dialog and save your changes to the Model Manager.

Remove a Skill

Removing a skill also removes its associations to any DEM Workers and workflows.

Prerequisites

Launch the vRealize Automation Designer console.

Procedure

1 On the ribbon, click **Manage Skills**.

2 In the Manage Skills dialog, select the name of the skill in the list on the left.

3 Click the **Remove** icon () at the top of the list of skills.

After you confirm that you want to delete the skill, its name appears dimmed to indicate that it is marked for deletion.

4 Click **OK** to close the Manage Skills dialog and save your changes to the Model Manager or **Cancel** if you do not want to delete the skill and its associations with DEMs and workflows.

CloudUtil Command Reference 5

This section provides a reference to the commands in the CloudUtil command line interface.

Note In the CloudUtil commands, the Model Manager is referred to as the `repository` and a Distributed Execution Manager (DEM) is referred to as an `agent`.

This chapter includes the following topics:

DEM Commands

The DEM commands enable you to view a list of Distributed Execution Managers registered with the Model Manager and add or remove associations between skills and DEMs.

DEM-Add-Skills

Associates skills with a registered Distributed Execution Manager.

Synopsis

```
CloudUtil.exe DEM-Add-Skills -n|--name <Name> -s|--skills <Skills> [--repository <Model Manager Root URI>] [-v|--verbose]
```

DEM-Add-Skills Arguments

Argument	Description
-n | - -name	Name of a registered Distributed Execution Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.

Argument	Description
-s | - -skills	Comma-delimited list of skills to associate with this Distributed Execution Manager.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Note The skills must already exist in the Model Manager. See "Skill-Install," on page 51.

DEM-List

Lists all Distributed Execution Managers registered with the Model Manager and their associated skills.

Synopsis

```
CloudUtil.exe DEM-List [--repository <Model Manager Root URI>] [-v|--verbose]
```

DEM-List Arguments

Argument	Description
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

DEM-Remove-Skills

Removes association between skills and a Distributed Execution Manager.

Synopsis

```
CloudUtil.exe DEM-Remove-Skills -n|--name <Name> -s|--skills <Skills> [--repository <Model
Manager Root URI>] [-v|--verbose]
```

DEM-Remove-Skills Arguments

Argument	Description
-n | - -name	Name of a registered Distributed Execution Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-s | - -skills	Comma-delimited list of skills to remove from this Distributed Execution Manager.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File Commands

The File commands enable you to store and manage files (typically scripts) in the Model Manager.

File-Export

Exports a file from the Model Manager.

Synopsis

```
CloudUtil.exe File-Export -n|--name <Name> -o|--output <Output File> [-i|--iteration
<Iteration>] [--repository <Model Manager Root URI>] [-v|--verbose]
```

File-Export Arguments

Argument	Description
-i | - -iteration	(Optional) Version string of the file in the Model Manager. Default is **0.0**.
-n | - -name	Friendly name of the file in the Model Manager.
-o | - -output	Path for file output.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File-Import

Imports a file into the Model Manager.

Synopsis

```
CloudUtil.exe File-Import -n|--name <Name> -f|--filename <File Name> [-d|--description
<Description>] [-i|--iteration <Iteration>] [--repository <Model Manager Root URI>] [-v|--
verbose]
```

File-Import Arguments

Argument	Description
-d | - -description	(Optional) Description of the file.
-f | - -filename	Path to a file to import into the Model Manager.
-i | - -iteration	(Optional) Version string of the file in the Model Manager. Default is **0.0**.
-n | - -name	Friendly name to assign to the file in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File-List

Lists all files imported into the Model Manager.

Synopsis

```
CloudUtil.exe File-List [--repository <Model Manager Root URI>] [-v|--verbose]
```

File-List Arguments

Argument	Description
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File-Remove-All

Removes all revisions for a given version of a file from the Model Manager.

Synopsis

```
CloudUtil.exe File-Remove-All -n|--name <Name> [-i|--iteration <Iteration>]
[--repository <Model Manager Root URI>] [-v|--verbose]
```

File-Remove-All Arguments

Table 5-1.

Argument	Description
-i | - -iteration	(Optional) Version string of the file in the Model Manager. Default is **0.0**.
-n | - -name	Friendly name of the file in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File-Remove-Rev

Removes a specific revision of a file from the Model Manager.

Synopsis

```
CloudUtil.exe File-Remove-Rev -n|--name <Name> -r|--revision <Revision> [-i|--iteration
<Iteration>] [--repository <Model Manager Root URI>] [-v|--verbose]
```

File-Export Arguments

Argument	Description
-i | - -iteration	(Optional) Version string of the file in the Model Manager. Default is **0.0**.
-n | - -name	Friendly name of the file in the Model Manager.
-r | - -revision	Revision of the file to remove.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File-Rollback

Reverts a file in the Model Manager to a specified revision.

Synopsis

```
CloudUtil.exe File-Rollback -n|--name <Name> -r|--revision <Revision> [-i|--iteration
<Iteration>] [--repository <Model Manager Root URI>] [-v|--verbose]
```

File-Rollback Arguments

Table 5-2.

Argument	Description
-i | - -iteration	(Optional) Version string of the file in the Model Manager. Default is **0.0**.
-n | - -name	Friendly name of the file in the Model Manager.
-r | - -revision	Revision of the file to revert to.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

File-Update

Updates a previously imported file in the Model Manager with a new revision.

Synopsis

```
CloudUtil.exe File-Update -n|--name <Name> -f|--filename <File Name> [-i|--iteration
<Iteration>] [--repository <Model Manager Root URI>] [-v|--verbose]
```

File-Update Arguments

Argument	Description
-f | - -filename	Path to the updated file.
-i | - -iteration	(Optional) Version string of the file in the Model Manager. Default is **0.0**.
-n | - -name	Friendly name of the file in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Operation Commands

The Operation commands enable you to manage custom operations in the Model Manager.

Operation-Create

Creates a custom operation or set of operations that can be performed on a machine based on an operation definition file.

Synopsis

```
CloudUtil.exe Operation-Create -c|--operationConfig <Operation Definition File> [--repository
<Model Manager Root URI>] [-v|--verbose]
```

Operation-Create Arguments

Argument	Description
-c | - -operationConfig	Path to an operation definition file (XML).
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Operation-Delete

Deletes a custom operation from the Model Manager.

Synopsis

```
CloudUtil.exe Operation-Delete -n|--name <Name> [--force] [--repository <Model Manager
Root URI>] [-v|--verbose]
```

Operation-Delete Arguments

Argument	Description
- -force	(Optional) Force deletion of the operation.
-n | - -name	Name of the custom operation in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Operation-List

Lists all custom operations in the Model Manager.

Synopsis

```
CloudUtil.exe Operation-List [--repository <Model Manager Root URI>] [-v|--verbose]
```

Operation-List Arguments

Argument	Description
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Skill Commands

The Skill commands enable you to manage the skills associated with Distributed Execution Managers and workflows.

Skill-Install

Installs a skill in the Model Manager.

Synopsis

```
CloudUtil.exe Skill-Install -n|--name <Name> [--repository <Model Manager Root URI>] [-v|--
verbose]
```

Skill-Install Arguments

Argument	Description
-n | - -name	Name for the skill in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Skill-List

List all skills installed in the Model Manager.

Synopsis

```
CloudUtil.exe Skill-List [--repository <Model Manager Root URI>] [-v|--verbose]
```

Skill-List Arguments

Argument	Description
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Skill-Uninstall

Uninstall a skill from the Model Manager.

Synopsis

```
CloudUtil.exe Skill-Uninstall -n|--name <Name> [--repository <Model Manager Root URI>]
[-v|--verbose]
```

Skill-Uninstall Arguments

Argument	Description
-n | - -name	Name of the skill to uninstall from the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Note A skill cannot be uninstalled if it is associated with either a Distributed Execution Manager or a workflow. See "DEM-Remove-Skills," on page 46 or "Workflow-Remove-Skills," on page 53.

Workflow Commands

The Workflow commands enable you to manage customizable IaaS workflows in the Model Manager, as well as the skills associated with any workflows.

Workflow-Add-Skills

Associate skills with a workflow in the Model Manager.

```
CloudUtil.exe Workflow-Add-Skills -n|--name <Name> -s|--skills <Skills> [--repository <Model
Manager Root URI>] [-v|--verbose]
```

Table 5-3. Workflow-Add-Skills Arguments

Argument	Description
Name	Name of a workflow in the Model Manager.
Skills	Comma-delimited list of skills to associate with this workflow.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Note The skills must already exist in the Model Manager. See "Skill-Install," on page 51.

Workflow-List

List all workflows installed in the Model Manager and their associated skills.

```
CloudUtil.exe Workflow-List [--repository <Model Manager Root URI>] [-v|--verbose]
```

Table 5-4. Workflow-List Arguments

Argument	Description
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Workflow-Remove-Skills

Removes association between skills and a workflow in the Model Manager.

Synopsis

```
CloudUtil.exe Workflow-Remove-Skills -n|--name <Name> -s|--skills <Skills> [--repository
<Model Manager Root URI>] [-v|--verbose]
```

Workflow-Remove-Skills Arguments

Argument	Description
-n | - -name	Name of a workflow in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-s | - -skills	Comma-delimited list of skills to remove from this workflow.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Workflow-Rollback

Reverts a workflow to a given revision.

Synopsis

```
CloudUtil.exe Workflow-Rollback -n|--name <Name> -r|--revision <Revision> [--repository <Model
Manager Root URI>] [-v|--verbose]
```

Workflow-Rollback Arguments

Argument	Description
-n | - -name	Name of the workflow in the Model Manager.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-r | - -revision	Revision of the workflow to revert to.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Workflow-Update

Update a customizable workflow with a new revision.

```
CloudUtil.exe Workflow-Update -f|--filename <File Name> -n|--name <Name> [-d|--description
<Description>] [--repository <Model Manager Root URI>] [-v|--verbose]
```

Table 5-5. Workflow-Update Arguments

Argument	Description
File Name	Path to a file (XAML) containing the updated workflow.
Name	Name of the workflow to update.
Description	(Optional) Description of workflow.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Import Commands

The import commands enable you to import one or more virtual machines into a vRealize Automation deployment.

Machine-BulkRegisterExport

Creates a CSV data file that is used to import virtual machines to a vRealize Automation deployment.

Synopsis

```
CloudUtil.exe Machine–BulkRegisterExport [–b|––blueprint] [–m|––managed] [–e|––exportNames] [–
p|––properties] –f|––filename <Value> [–g|––group <Value>] [–i|––ignore] [–o|––owner <Value>] [–
t|––machinetype <Value>] [–n|––resourceName <Value>] [–r|––resourceType <Value>] [––repository
<Value>] [–sn|––sourcename <Value>] [–st|––sourcetype <Value>] –u|––user <value> [–v|––verbose]
```

Machine-BulkRegisterExport Arguments

Table 5-6.

Argument	Description
-b | - -blueprint	(Optional) Include blueprint name.
-e | - -exportNames	(Optional) Export names instead of GUIDs.
-f | - -filename	Specify the name of the CSV data file containing a list of machine names, for example, `filename.csv`. File is saved in the current path by default. You can also specify the complete path to a preferred directory.
-g | - -group	(Optional) Specify business group name, for example, Engineering.
-i | - -ignore	(Optional) Ignore invalid arguments.
-m | - -managed	(Optional) Export managed virtual machines. The default is export unmanaged virtual machines.
-n | - -resourceName	(Optional) To filter by resource name, specify the name of the Compute Resource or Endpoint.
-o | - -owner	(Optional) Specify owner of imported virtual machine, for example, jsmith.
-p | - -properties	(Optional) Export properties for managed virtual machines.
-r | - -resourceType	(Optional) To filter by resource type, specify 1 for Compute Resource or 2 for Endpoint .

Table 5-6. (Continued)

Argument	Description
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-sn | - -sourcename	(Optional) Specify the name of the cluster or endpoint .
-st | - -sourcetype	(Optional) Specify the source type as Cluster or Endpoint .
-t | - -machinetype	(Optional) Specify machine type to be exported, for example, Virtual, Physical, Cloud, AppService, vApp.
-u | - -user	Specify the Fabric Administrator who performs the bulk registration.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.

Machine-BulkRegisterImport

Imports one or more virtual machines to a target vRealize Automation deployment.

Synopsis

```
CloudUtil.exe Machine–BulkRegisterImport [–b|––batch][–d|––delay <value>] –f|––filename <value>
[–i|––ignore] [–h|––humanreadable] –n|––name <value> [––repository <value>] [–s|––skipUser] –t|––
time <value> –u|––user <value> [–v|––verbose] [–w|––whatIf]
```

Machine-BulkRegisterImport Arguments

Table 5-7.

Argument	Description
-b | - -batch	(Optional) Batch size
-d | - -delay	(Optional) Specify processing delay time in this format: hh:mm:ss, for example, 02:20:10.
-f | - -filename	Specify the CSV data file name containing the list of machine names. For example, `filename.csv`.
-h | --humanreadable	(Optional) Input file contains the virtual machine names and not the GUIDs.
-i | - -ignore	(Optional) Ignore registered or managed virtual machines.
-n | - -name	Specify the name of the work queue to perform the import to the target vRealize Automation.
- -repository	(Optional) The root URI of the Model Manager, for example, http://*hostname*/repository. Default is specified in the CloudUtil config file in the repositoryAddress key under the <appSettings> section.
-s | - -skipUser	(Optional) Sets a machine's owner to the value listed in the Owner column of the CSV data file without verifying that the user exists. Selecting this option can decrease the time required for import.
-t | - -time	Specify the workflow start time in the format MM/DD/YYYY hh:mm GMT, for example, 04/18/2014 10:01 GMT. The specified start time is assumed to be the server's local time and not the local time of the user's workstation.

Table 5-7. (Continued)

Argument	Description
-u | - -user	Specify the Fabric Administrator who performs the bulk registration.
-v | - -verbose	(Optional) If an error occurs, outputs a stack trace instead of just the exception message.
whatif	(Optional) Set to validate the CSV file but do not import any virtual machines.

vRealize Automation Workflow Activity Reference

6

VMware provides a library of workflow activities with vRealize Automation Designer for use in customizing workflows.

Five categories of Windows Workflow Foundation activities are also included in vRealize Automation Designer, including Control Flow, Flowchart, Primitives, Collection and Error Handling.

This section provides a reference to the IaaS workflow activities included with vRealize Automation Designer in the `DynamicOps.Repository.Activities` and `DynamicOps.Cdk.Activities` namespaces. Activities related to calling vCenter Orchestrator workflows are described in "Using vRealize Orchestrator Workflow Activities," on page 32.

Note In the IaaS activity library, the Model Manager is referred to as the `repository`.

This chapter includes the following topics:

DynamicOps.Repository.Activities

The `DynamicOps.Repository.Activities` namespace contains basic workflow activities for IaaS workflows.

AddLink

Adds the specified link to the set of objects the `DataServiceContext` is tracking.

Table 6-1. AddLink Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` to which to add the link.
Source	Object	The source object for the new link.
SourceProperty	String	The name of the navigation property on the source object that returns the related object.
Target	Object	The object related to the source object by the new link.

AddObject

Adds the specified object to the set of objects the `DataServiceContext` is tracking.

Table 6-2. AddObject Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` to which to add the object.
Instance	Object	The object to be tracked by the `DataServiceContext`.

AttachTo

Notifies the `DataServiceContext` to start tracking the specified resource.

Table 6-3. AttachTo Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` that should track the resource.
Instance	Object	The resource to be tracked by the `DataServiceContext`. The resource is attached in the Unchanged state.

CreateRepositoryServiceContext<T>

Creates a context for a model loaded into the Model Manager.

When you add this activity to a workflow in vCloud Automation Center Designer, you must select a class that inherits from the `RepositoryServiceContext` class.

Table 6-4. CreateRepositoryServiceContext<T> Activity Input Parameters

Argument	Type	Description
Uri	URI	(Optional) Root URI to use in connecting to the model.
Username	String	(Optional) Username to use in connecting to the context.

Table 6-5. CreateRepositoryServiceContext<T> Activity Output Parameters

Argument	Type	Description
Result	RepositoryServiceContext	The specific type returned is an instance of the class selected when the activity was added to the workflow.

DeleteLink

Changes the state of the link to deleted in the list of links being tracked by the `DataServiceContext`.

Table 6-6. DeleteLink Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` from which to delete the link.
Source	Object	The source object in the link to be marked for deletion.
SourceProperty	String	The name of the navigation property on the source object that is used to access the target object.
Target	Object	The target object involved in the link that is bound to the source object. The target object must be of the type identified by the source property or a subtype.

DeleteObject

Changes the state of the specified object to be deleted in the `DataServiceContext`.

Table 6-7. DeleteObject Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` from which to delete the resource.
Instance	Object	The tracked entity to be changed to the deleted state.

InvokeRepositoryWorkflow

Executes a workflow installed in the Model Manager.

Table 6-8. InvokeRepositoryWorkflow Activity Input Parameters

Argument	Type	Description
WorkflowType	WorkflowDefinition entity	The workflow to execute.
WorkflowInputs	Dictionary<string, object>	(Optional) Inputs to the workflow.
CallingInstance	WorkflowInstance entity	(Optional) The workflow that calls the executed workflow and to which it will return.

LoadProperty

Loads deferred content for a specified property from the data service.

Table 6-9. LoadProperty Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` from which to load the property.
Instance	Object	The entity that contains the property to load.
InstanceProperty	String	The name of the property of the specified entity to load.

SaveChanges

Saves the changes that the `DataServiceContext` is tracking to storage.

Table 6-10. SaveChanges Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` that is tracking the changes to save.

SetLink

Notifies the `DataServiceContext` that a new link exists between the objects specified and that the link is represented by the property specified in the `SourceProperty` argument.

Table 6-11. SetLink Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` to notify of the link.
Source	Object	The source object for the new link.
SourceProperty	String	The property on the source object that identifies the target object of the new link.
Target	Object	The child object involved in the new link to initialize by calling this method. The target object must be a subtype of the type identified by SourceProperty. If Target is set to null, the call represents a delete link operation.

UpdateObject

Changes the state of the specified object in the `DataServiceContext` to Modified.

Table 6-12. UpdateObject Activity Input Parameters

Argument	Type	Description
DataServiceContext	RepositoryServiceContext	The `DataServiceContext` tracking the entity to update.
Instance	Object	The tracked entity to be assigned to the Modified state.

DynamicOps.Cdk.Activities

The `DynamicOps.Cdk.Activities` namespace contains advanced activities for IaaS workflows.

ExecutePowerShellScript

Executes a PowerShell script stored in the Model Manager under the specified name.

Before you use the `ExecutePowerShellScript` activity, you must first load the script that you want to execute into the Model Manager using the `CloudUtil File–Import` command.

Table 6-13. ExecutePowerShellScript Activity Input Parameters

Argument	Type	Description
ScriptName	String	Name in the Model Manager of the script to execute.
ScriptVersion	Object	(Optional) Version in the Model Manager of the script to execute. Default is 0.0.
MachineId	Guid	(Optional) If specified, the machine is loaded and all its properties are passed to the script.

Table 6-13. ExecutePowerShellScript Activity Input Parameters (Continued)

Argument	Type	Description
Arguments	Dictionary<string,string>	Additional arguments to pass to the script. If MachineId is specified and there is a machine property with the same name as an argument (case-insensitive), the value of the machine property overrides the value of the argument.
PSModules	IEnumerable<string>	(Optional) Modules loaded into the PowerShell runtime during command execution. This option is only available in the Properties pane and not in the Designer pane.

Table 6-14. ExecutePowerShellScript Activity Output Parameters

Argument	Type	Description
Output	Collection<PSObject>	Output of script if any. Throws exception on error.

If you receive the error message `Type PSObject is not defined` in the vRealize Automation Designer console when you are dealing with the output of `ExecutePowerShellScript`, perform the following steps:

1 Click **Imports** in the lower left corner of the Designer pane.

2 Select the **System.Management.Automation** assembly.

ExecuteSshScript

Executes an SSH script stored in the model manager under the specified name.

Before you use the `ExecuteSshScript` activity, you must first load the script that you want to execute into the Model Manager using the `CloudUtil File-Import` command.

Table 6-15. ExecuteSshScript Activity Input Parameters

Argument	Type	Description
ScriptName	String	Name in the Model Manager of the script to execute.
Host	String	Server name against which to execute the script.
Username	String	Username to use in connecting to the host.
Password	String	Password to use in connecting to the host.
ScriptVersion	Object	(Optional) Version in the Model Manager of the script to execute. Default is 0.0.
Timeout	TimeSpan	(Optional) Period of time after which execution of the script times out. Default is 30 minutes.

Table 6-16. ExecuteSshScript Activity Output Parameters

Argument	Type	Description
EnvironmentVariables	Dictionary<string, string>	Script execution result if any.

GetMachineName

Gets a machine's name.

Table 6-17. GetMachineName Activity Input Parameters

Argument	Type	Description
MachineId	Guid	The machine whose name to retrieve.

Table 6-18. GetMachineName Activity Output Parameters

Argument	Type	Description
MachineName	String	Name of the machine identifed by `MachineId`.

GetMachineOwner

Gets the username of a machine's owner.

Table 6-19. GetMachineOwner Activity Input Parameters

Argument	Type	Description
MachineId	Guid	The machine whose owner to retrieve.

Table 6-20. GetMachineOwner Activity Output Parameters

Argument	Type	Description
Owner	String	Owner of the machine identified by `MachineId`, or null if there is no owner.

GetMachineProperties

Gets the list of custom properties associated with a machine.

Table 6-21. GetMachineProperties Activity Input Parameters

Argument	Type	Description
MachineId	Guid	The machine whose properties to retrieve.

Table 6-22. GetMachineProperties Activity Output Parameters

Argument	Type	Description
Properties	Dictionary<string, string>	List of the machine's properties. Values are returned decrypted if they were stored encrypted.

GetMachineProperty

Gets the value of the specified property for a machine.

Table 6-23. GetMachineProperty Activity Input Parameters

Argument	Type	Description
MachineId	Guid	The machine from which to retrieve a property.
PropertyName	String	Name of the property whose value to return.
IsRequired	bool	If the property is required and is not found the activity throws an exception, otherwise returns null.

Table 6-24. GetMachineProperty Activity Output Parameters

Argument	Type	Description
PropertyValue	String	Value of the property specified by `PropertyName`. The value is returned decrypted if it was stored encrypted.

GetScriptFromName

Gets the contents of the script stored in the Model Manager under the specified name.

Table 6-25. GetScriptFromName Activity Input Parameters

Argument	Type	Description
ScriptName	String	Name in the Model Manager of the script to retrieve.
ScriptVersion	Object	(Optional) Version in the Model Manager of the script to retrieve. Default is 0.0.

Table 6-26. GetScriptFromName Activity Output Parameters

Argument	Type	Description
ScriptContent	String	Contents of the script identified by `ScriptName`.

InvokePowerShell

Executes a PowerShell command.

Table 6-27. InvokePowerShell Activity Input Parameters

Argument	Type	Description
CommandText	String	Command to execute.
Arguments	IEnumerable<string>	(Optional) Arguments to the command.
Input	IEnumerable	(Optional) The input pipeline.
IsScript	bool	(Optional) Indicates whether `CommandText` is a script. Default is False. This option is only available in the Properties pane and not in the Designer pane.
Parameters	Collection	(Optional) Collection of name-value pairs passed as parameters to the PowerShell script. This option is only available in the Properties pane and not in the Designer pane.
PowerShellVariables	Collection	(Optional) Variables copied into the PowerShell runtime. This option is only available in the Properties pane and not in the Designer pane.
PSModules	IEnumerable<string>	(Optional) Modules loaded into the PowerShell runtime during command execution. This option is only available in the Properties pane and not in the Designer pane.
Runspace	Runspace	(Optional) Creating a PowerShell runspace and supplying it to this argument enables you to reuse the same runspace in multiple PowerShell invocations, which may result in performance improvements. This option is only available in the Properties pane and not in the Designer pane.

Table 6-28. InvokePowerShell Activity Output Parameters

Argument	Type	Description
Output	Collection<PSObject>	Output of command if any. Throws exception on error.
Errors	Collection<ErrorRecord>	Errors resulting from execution if any.

If you receive the error message `Type PSObject is not defined` in the vRealize Automation Designer console when you are dealing with the output of `ExecutePowerShellScript`, perform the following steps:

1 Click **Imports** in the lower left corner of the Designer pane.

2 Select the **System.Management.Automation** assembly.

InvokeSshCommand

Executes an SSH command.

Table 6-29. InvokeSshCommand Activity Input Parameters

Argument	Type	Description
CommandText	String	Command to execute.
Host	String	Server name against which to execute the command.
Username	String	Username to use in connecting to the host.
Password	String	Password to use in connecting to the host.
Timeout	TimeSpan	(Optional) Period of time after which execution of the command times out. Default is 30 minutes.

Table 6-30. InvokeSshCommand Activity Output Parameters

Argument	Type	Description
EnvironmentVariables	Dictionary<string, string>	Output of command if any. Throws exception on error.

LogMachineEvent

Logs a machine event to the user log that is visible to the machine owner.

Table 6-31. LogMachineEvent Activity Input Parameters

Argument	Type	Description
MachineId	Guid	Machine generating the event to log.
Message	String	Message to write to the user log.
Type	String	Select a message type from the drop-down list (Info, Warn, Error)

LogMessage

Logs to the Distributed Execution Manager log.

Table 6-32. LogMessage Activity Input Parameters

Argument	Type	Description
Message	String	Message to write to the DEM log.
MessageCategory	String	Select a category from the drop-down menu (**Debug**, **Error**, **Info**, **Trace**) or enter a custom category.
MessageSeverity	String	Select a severity from the drop-down menu; bound to the list of Severities supplied in `System.Diagnostics.TraceEventType`.

RunProcess

Executes a process on the same machine as the DEM that executes this activity.

Note vRealize Automation cannot present the UI from processes launched by the `RunProcess` activity to the user, therefore these process must be non-interactive. In order to avoid leaving orphaned processes on the DEM machine, the processes must also be self-terminating.

Table 6-33. RunProcess Activity Input Parameters

Argument	Type	Description
Command	String	Path to the executable to run on the DEM machine.
WorkingDirectory	String	(Optional) The working directory under which the process should run.
Arguments	String	(Optional) The list of command-line arguments to pass to the command.
WaitForExit	bool	(Optional) If true, the workflow waits for the process to complete before continuing with the workflow. Default is false. This option is only available in the Properties pane and not in the Designer pane.

SendEmail

Sends an email to the given set of addresses.

Table 6-34. SendEmail Activity Input Parameters

Argument	Type	Description
To	IEnumerable<string>	The list of addresses to which to send the email.
From	String	The address with which to populate the “From” field of the email.
Subject	String	The subject line for the email.
Body	String	The body text of the email.
Host	String	The host name or IP address of the outgoing SMTP server.
Port	Integer	The SMTP port on the server specified in Host. This option is only available in the Properties pane and not in the Designer pane.
CC	IEnumerable<string>	(Optional) The address or list of addresses to copy on the email. This option is only available in the Properties pane and not in the Designer pane.
Bcc	IEnumerable<string>	(Optional) The address or list of addresses to blind copy on the email. This option is only available in the Properties pane and not in the Designer pane.
EnableSsl	bool	(Optional) Indicates whether to use SSL. This option is only available in the Properties pane and not in the Designer pane.

Table 6-34. SendEmail Activity Input Parameters (Continued)

Argument	Type	Description
UserName	String	The user name with which to authenticate with the SMTP server specified in `Host` . This option is only available in the Properties pane and not in the Designer pane.
Password	String	The password of the user specified in `UserName`. This option is only available in the Properties pane and not in the Designer pane.

SetMachineProperty

Creates or updates a custom property on a machine.

Table 6-35. SetMachineProperty Activity Input Parameters

Argument	Type	Description
MachineId	Guid	Machine on which to create or update the custom property.
PropertyName	String	Name of property to create or update.
PropertyValue	String	Value with which to create or update the property.
IsEncrypted	bool	(Optional) Indicates whether the value of the property is encrypted.
IsHidden	bool	(Optional) Indicates whether the property is a hidden property.
IsRuntime	bool	(Optional) Indicates whether the requesting user provides the property value at request time (equivalent to being marked Prompt User in the vRealize Automation console).

SetWorkflowResult

Sets an external workflow's state to either Complete or Failed to be honored by `ExternalWF.xml` settings.

Table 6-36. SetWorkflowResult Activity Input Parameters

Argument	Type	Description
WorkflowId	Guid	The workflow for which to set the state.
Next State	WorkflowState	Select **Complete** or **Failed** from the drop-down menu.

Index