

HP Cloud Service Automation

Software Version 4.20



Topology Components Guide

© Copyright 2015 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Restricted rights legend: Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Adobe® is a trademark of Adobe Systems Incorporated. Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. RED HAT READY™ Logo and RED HAT CERTIFIED PARTNER™ Logo are trademarks of Red Hat, Inc.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation in the United States and other countries, and are used with the OpenStack Foundation's permission.

Contents

| | |
|--|-----------|
| Overview | 4 |
| HP CSA Content Sources | 5 |
| Out-of-the-box (OOTB) content | 5 |
| HP Operations Orchestration | 5 |
| HP Server Automation | 5 |
| Chef | 6 |
| Content Provided by HP Operations Orchestration | 6 |
| HP CSA – HP OO Integration | 6 |
| HP OO Server Setup | 6 |
| Standard versus Custom Content | 6 |
| Standard HP OO Content Guidelines | 7 |
| Folder Structure | 7 |
| Provider Type | 7 |
| Component Type | 8 |
| Version | 8 |
| Flows | 8 |
| Input and Output Properties | 9 |
| HP OO Content Import | 11 |
| Import Wizard | 11 |
| Content Source | 12 |
| Content Selection | 13 |
| Parameter Mapping | 13 |
| Post-Import Adjustments | 15 |
| HP OO Import Limitations | 15 |
| HP OO Operations | 15 |
| Single Flow Operations | 15 |
| Unique Lifecycle Operations | 15 |
| Content Provided by Chef | 16 |
| Supported Configurations | 17 |
| Prerequisites | 17 |
| Chef server Installation | 17 |
| Chef client Installation | 18 |

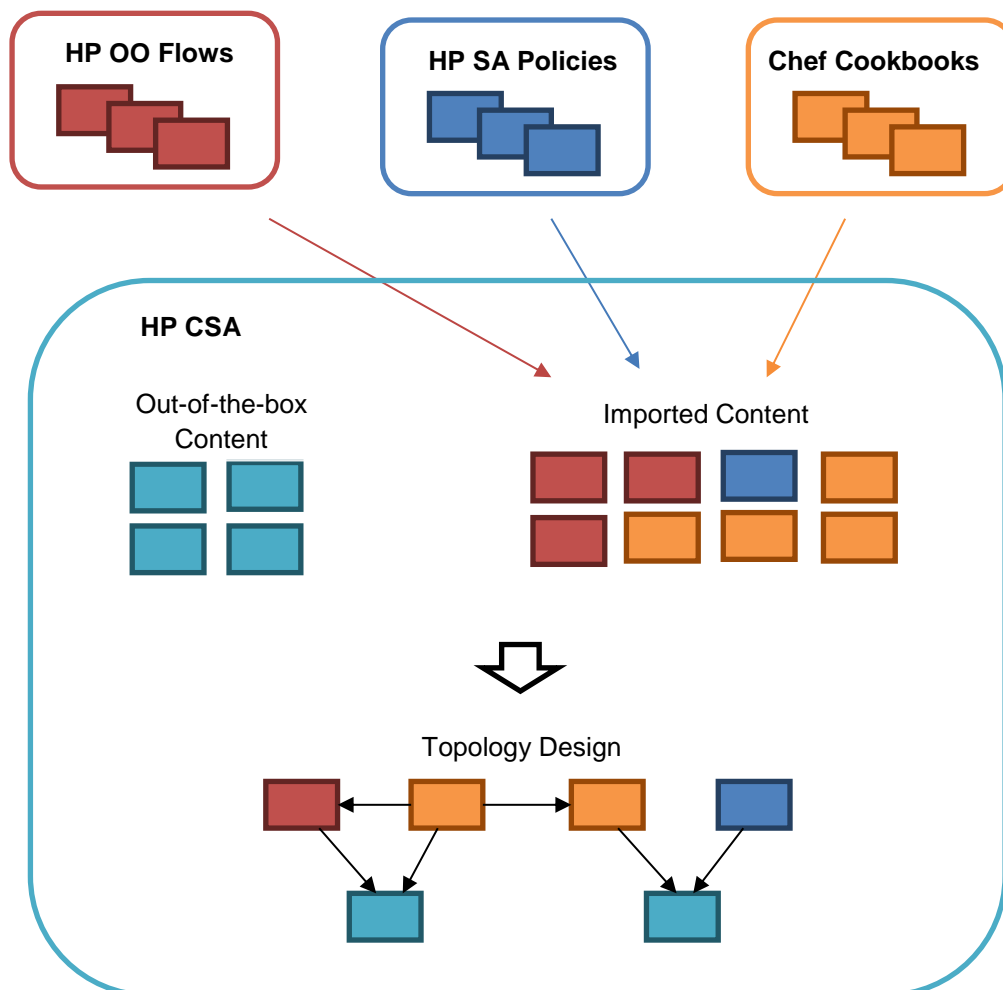
| | |
|--|-----------|
| Required User Accounts | 19 |
| Import Chef Cookbooks | 19 |
| SSH setup..... | 20 |
| VM Template Working with Chef | 20 |
| Steps to Create VM Template | 20 |
| Customization Specification | 21 |
| The Chef Provider in HP CSA | 21 |
| Import Cookbooks as Components..... | 22 |
| Content Source | 22 |
| Content Selection | 23 |
| Recipes in a Cookbook..... | 23 |
| Modification of Chef Components | 24 |
| Testing and Debugging Chef Components | 24 |
| Fine-Tuning the Imported Content..... | 24 |
| Component Overview | 25 |
| Server Capability | 26 |
| Lifecycle Operations | 26 |
| Custom Operations..... | 28 |
| Component Properties..... | 28 |
| Relationships to Other Components..... | 29 |
| Passing Values between Components | 29 |
| Testing and Debugging | 32 |
| Modification Scenarios..... | 32 |
| Configuring Modifiable Properties..... | 35 |
| Terminology..... | 36 |
| For More Information | 38 |

Overview

Most IT tasks, such as provisioning VMs, software installation, or registering users into a system, can be automated. HP Cloud Service Automation (HP CSA) is the right tool for that. The HP CSA Topology Designer provides your organization with a simple, scalable way to design complex cloud topology. Tasks that would normally take hours can take minutes with HP CSA's innovative approach to service design.

With support for infrastructure providers such as HP Helion OpenStack, VMware, Amazon, or Chef, as well as many HP enterprise products, such as SiteScope and Server Automation, you have the tools and integrations to meet your service needs.

HP CSA design components execute the set of IT tasks required for infrastructure or platform provisioning using a simple lifecycle: deploy, manage, and undeploy. In addition, HP CSA can import design components from supported providers. For example, in the case of a Chef provider, HP CSA can import Chef cookbooks into CSA design components. Imported components are then modeled into a declarative topology design.



Each topology design consists of *components* connected by *relationships*. Components represent the final shape of the deployment, and the relationships represent the dependencies between the components. The designer user declares *what* should be done, then HP CSA figures out *how* it should be done. The sequence of the tasks is automatically computed based on designed dependencies among components.

Once a topology design is created, HP CSA generates a proper execution plan consisting of the tasks required to get the design deployed. HP CSA uses HP Operations Orchestration (HP OO) to process the execution plan, which is represented by an HP OO *master-flow*. The HP OO master-flow consists of steps representing the deployment of each component. The details behind each individual component can be customized with your specific business logic.

HP CSA works with various kinds of content, including components available in HP CSA out-of-the-box. These are infrastructure components (like server or network) that can be provisioned by VMware vCenter, Amazon EC2 or HP Helion OpenStack.

Besides the out-of-the-box components, HP CSA can load content from other systems, representing this content as components, and then use these imported components in topology designs. Currently, HP OO flows, HP Server Automation policies and Chef cookbooks are supported.

HP CSA Content Sources

HP CSA topology designs are composed of components, including those from the sources below.

Out-of-the-box (OOTB) Content

This is the content present in HP CSA after installation (VMware vCenter, Amazon EC2, HP Helion OpenStack components).

HP Operations Orchestration

HP Operations Orchestration (HP OO) links automated tasks into flows. HP CSA can then wrap the result of a flow execution as a component, which can be used as a building block for more complex designs.

HP CSA uses HP OO in several ways:

- As a process executor for sequenced designs.
- As a process executor for topology designs, where the HP OO master-flow is the implementation of the design execution plan.
- As a component import source, where new topology components can be imported from existing HP OO flows. (For more information, see [Content Provided by HP Operations Orchestration](#) later in this document.)

HP Server Automation

HP Server Automation (HP SA) provides complete automated lifecycle management for enterprise servers. HP SA automates the deployment of applications using software policies, providing a proven, scalable, and heterogeneous solution across physical and virtual servers (including VM templates).

The HP SA policies can be used in HP CSA for application deployment on infrastructure provisioned by VMware vCenter. The software policies are wrapped into HP CSA components, which can be used in a topology design. During fulfillment of the design, HP SA deploys the associated policies on the infrastructure. For more information about importing HP SA components, see the *Import Components* topic in the HP CSA Management Console online help.

Chef

Chef is an automation framework that deploys servers and applications to any physical, virtual, or cloud location. Chef operates with *cookbooks* consisting of *recipes*.

HP CSA leverages Chef cookbooks for installation of software on top of infrastructure provisioned by providers such as VMware vCenter or Amazon EC2. The Chef cookbooks can be wrapped into HP CSA components that can be used for a topology design composition.

Content Provided by HP Operations Orchestration

HP Operations Orchestration (HP OO) serves as a content provider for HP CSA. The HP OO flows can be imported in CSA as components to form complex topology designs.

HP CSA – HP OO Integration

HP CSA comes with a number of out-of-the-box components. Many of them rely on integration with HP OO; therefore, HP CSA must be configured after installation so that it communicates with a running OO server.

HP OO Server Setup

For HP OO server installation, refer to the HP Operations Orchestration documentation available at <https://softwaresupport.hp.com/>

To configure HP OO for HP CSA, log into the HP OO Central web interface to create an *admin* user:

- Username: admin
- Password: cloud
- Roles: ADMINISTRATOR

Standard versus Custom Content

HP CSA is able to create components automatically by importing existing HP OO content, specifically HP OO flows. To enable this feature, HP CSA expects content to follow a specific naming and structure convention. This convention is described in the guidelines below.

HP OO content following the guidelines is called *standard content*. Note that HP CSA is capable of importing almost any existing content even if it does not follow the guidelines. Such HP OO content is called *custom content*.

Standard HP OO Content Guidelines

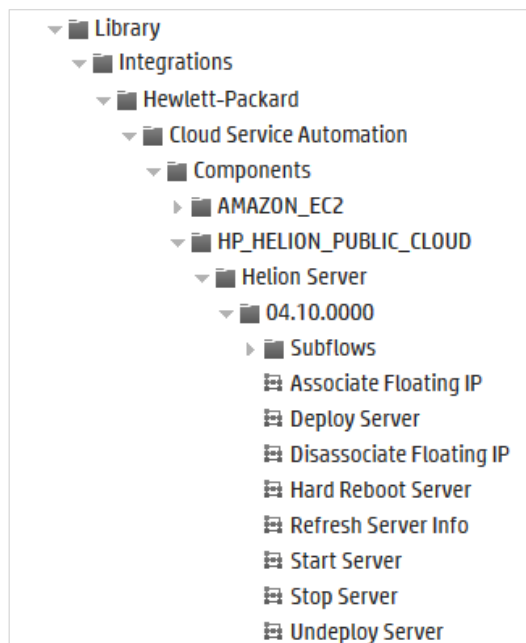
These instructions are for developers who develop content in HP OO and want to import that content into HP CSA.

IMPORTANT: A *component* is built from a set of flows. These flows implement the component operations. At a minimum there must be a flow for deployment or provisioning.

Folder Structure

The folder structure is important, and all flows representing a component must be placed in one folder. In the content pack or on the HP OO server, the flows must be stored in a path, as shown on the right. For example:

```
Library > Integrations > Hewlett-Packard > Cloud
Service Automation > Components > $ProviderTypeName >
$ComponentType > $Version > @Flows
```

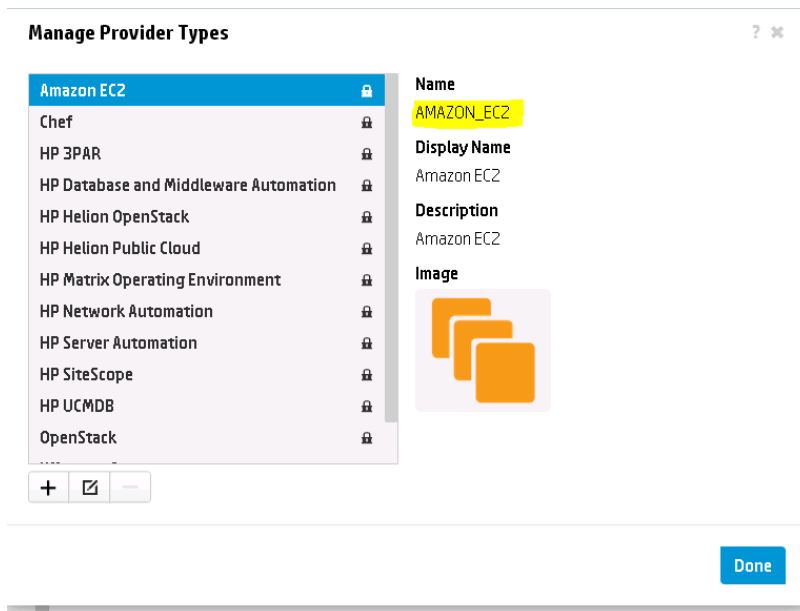


The values beginning with \$ are placeholders for the **provider type name**, the **component type**, and the **version**. The *@Flows* value represents a list of individual flows in the folder. Optionally, there can be a *Subflows* folder to gather additional entities used in the flows.

Provider Type

HP CSA includes out-of-the-box (OOTB) provider types. You can also create your own provider types. The provider type name values are inferred from the provider type label. If you create a **Custom Provider Type**, the name must be in uppercase with underscores; for example, CUSTOM_PROVIDER_TYPE. The uppercase provider type name must be used as the folder name for standard content. The provider type name value is also available in the Provider Management area of the HP CSA Cloud Service Management Console, as shown below.

| Provider Type Names |
|------------------------|
| AMAZON_EC2 |
| CHEF |
| HP_3PAR |
| HP_DMA |
| HP_CLOUDOS |
| HP_HELION_PUBLIC_CLOUD |
| HP_MOE |
| HP_NETWORK_AUTOMATION |
| HP_SA |
| HP_SITESCOPE |
| HP_UCMDB |
| OPENSTACK |
| VMWARE_VCENTER |



Component Type

The *\$ComponentType* is a placeholder for the component type name, which appears as one of the folders in the flow path used for standard content. In HP CSA, the folder name is used as the component type name. Note that that name can be changed later when editing the component. Keep in mind that the higher-level, provider type folder (the parent) does not affect the component name. For example, suppose you have a folder structure like this:

```
Library > Integrations > Hewlett-Packard > Cloud Service Automation > OPENSTACK -> Server -> 1.0
```

In this case, the component is called *Server*. If you follow this naming approach, you could end up with many *Server* components in the tree structure. Furthermore, the component name together with version provides unique identification of the component. HP CSA does not let you import multiple components named *Server* if they have the same version, regardless of the related provider type. Consider using more descriptive names for a component, such as *OpenStack Server* to avoid naming conflicts.

Version

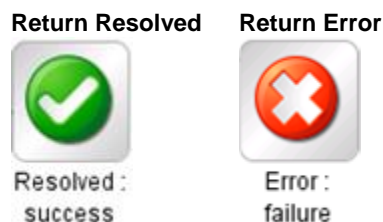
HP CSA does not provide component version-management capabilities. However, importing versioned standard content is supported. A component is created with a version matching the leaf folder in the flow path. This component version, along with name, uniquely identifies the component to HP CSA.

Flows

The individual flows should be placed under the version folder. The name of the flow is used automatically as an operation name in the new component. The flow name is important for recognition of the lifecycle phase related to the operation.

Each HP OO flow should have an input property defined as a constant called *LIFECYCLE_PHASE*. The property does not relate to the flow logic. It is used as meta-information, marking the flow's purpose related to the component lifecycle.

NOTE: Every HP OO flow used as an action in a component must have both Success and Failure indicators as a possible outcome:



As you can see below, there are duplicate indicators of the operation lifecycle phase (the duplication will be removed in a future release). Both the prefix of flow name and *LIFECYCLE_PHASE* input property must be set appropriately. Currently, the flow name and the value of the constant property should be set as follows:

| Flow Purpose | Flow Name Prefix | LIFECYCLE_PHASE |
|--|---------------------------|---------------------|
| Deployment | <i>Deploy or Create</i> | deploying |
| Undeployment | <i>Undeploy or Delete</i> | undeploying |
| Modification | <i>Modify</i> | modifying |
| Undo a successful modification | <i>Unmodify</i> | unmodifying |
| Handle a failure during deployment | | deploying_failure |
| Handle a failure during undeployment | | undeploying_failure |
| Handle a failure during modification | <i>Modify Failure</i> | modifying_failure |
| Custom public action executable on a deployed instance | | deployed |

Input and Output Properties

The HP OO server automatically wraps all flows with a **result** output property; however, HP suggests that you define flow outputs as explicitly as output properties.


- **response** – Mandatory property. Each HP OO flow used in a component operation should have a **response** output property. HP CSA relies on the **response** property value to determine the state of the execution of the respective component.
 - For deployment, undeployment flows and flows related to public actions, the value must be either **success** or **failure** based on the outcome of the flow.
 - For modify flows, possible values for the response property are **success**, **noop** or **failure**:
 - **Success** indicates that the attempted modification was successful.
 - **Noop** indicates that no action was taken that would affect the component state or properties.
 - **Failure** indicates that the attempted modification was unsuccessful.

- Unmodify flows should overwrite the **response** property, setting this property to **failure** if the unmodify failed. Unmodify flows should overwrite the response property to **noop** if unmodify succeeded, indicating that the component state and property values are unaffected due to modify transition.
- The modify failure flow should set the **response** property to **failure always** to indicate failure to modify this component

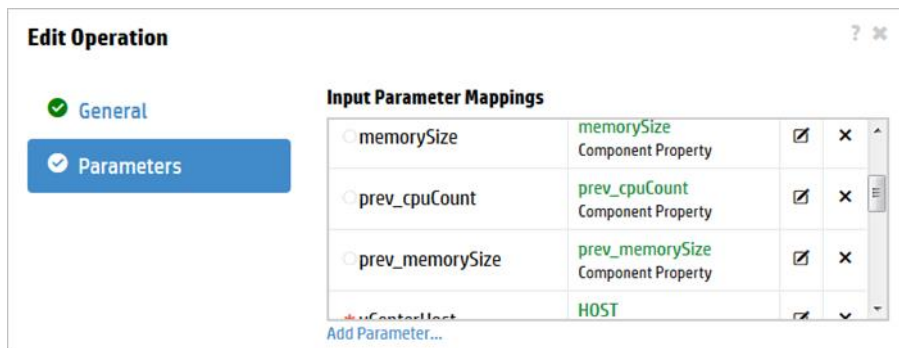
NOTE: If you include the deploy failure handler and/or undeploy failure handler, make sure that the **response** property value is set to **failure**.

- **result** – Optional property. Each HP OO flow used in a component operation should have a **result** output property. This property holds contextual messages; for example, detailed information about failure.

HP CSA supports the modification of properties on an active subscription. HP CSA passes previous property values to modification HP OO flows if the flows explicitly request these values by defining an additional flow input with a **prev_** prefix. For example, if the modification HP OO flow defines an input of **memorySize** and requires its previous value, you must also define an input of **prev_memorySize**.

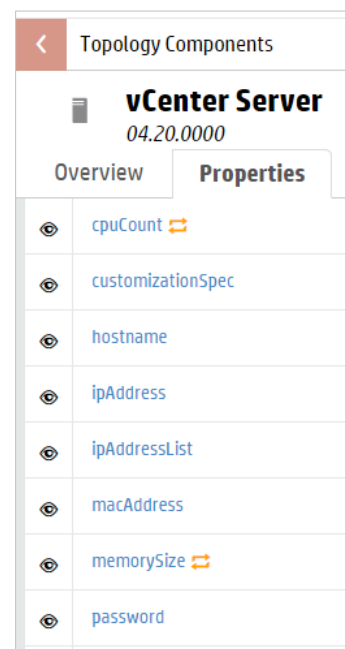
The graphic on the right shows the **cpuCount** and **memorySize** properties, and also shows how the flow has requested the previous values of these properties. This request is denoted by an icon  next to the property name in the HP CSA Topology Components view.



IMPORTANT: The **prev_** HP OO flow inputs **do not** display as properties on the component; only their un-prefixed counterparts show and are appropriately marked. However, the **prev_** HP OO flow inputs **do** display on an Operation's Input Parameter Mapping, as shown below. These mappings are used by the modification operation and **MUST NOT** be edited in the Component Editor.



| Input Parameter Mappings | | | |
|---------------------------------------|---------------------------------------|-------------------------------------|----------------------------------|
| <input type="radio"/> memorySize | memorySize Component Property | <input checked="" type="checkbox"/> | <input type="button" value="X"/> |
| <input type="radio"/> prev_cpuCount | prev_cpuCount Component Property | <input checked="" type="checkbox"/> | <input type="button" value="X"/> |
| <input type="radio"/> prev_memorySize | prev_memorySize Component Property | <input checked="" type="checkbox"/> | <input type="button" value="X"/> |
| <input type="radio"/> vCenterHost | HOST | <input checked="" type="checkbox"/> | <input type="button" value="X"/> |

Add Parameter...



| vCenter Server 04.20.0000 | |
|------------------------------|--|
| Overview | Properties |
| <input type="radio"/> | cpuCount  |
| <input type="radio"/> | customizationSpec |
| <input type="radio"/> | hostname |
| <input type="radio"/> | ipAddress |
| <input type="radio"/> | ipAddressList |
| <input type="radio"/> | macAddress |
| <input type="radio"/> | memorySize  |
| <input type="radio"/> | password |

A component's **modify** flow may communicate the status of the modification, an error code, or any other useful information to the **modify failure** flow by placing this information in an output field called **modifyReturnValue**. This value may be received by the **modify failure** flow in an input field called **modifyFailureValue**, allowing the **modify failure** flow to clean up any failure in the **modify** flow intelligently. This is especially useful for complex, multi-step flows when a failure might have occurred at any step, which can be indicated by returning specific error code from the **modify** flow via **ModifyReturnValue**.

IMPORTANT: If these fields are set on the HP OO flows, the mapping between them is automatically handled within HP CSA and **MUST NOT** be modified from the Component Editor.

HP OO Content Import

This chapter describes the process of creating HP CSA components from HP OO flows, covering both standard and custom OO content. The OO content is imported into HP CSA and then encapsulated into components. These components can be used in topology designs, but typically require a bit of manual adjustment and fine-tuning.

All HP CSA components have **properties**, **operations** and **relationships**.

- Property values are the inputs that parameterize the component realization. Properties can also hold results from the realization that will be displayed or even used by other components.
- Operations are of two kinds:
 - Lifecycle operations are coupled with the component deployment and undeployment.
 - Custom operations can later be exposed as public actions that a Subscriber can execute in the Marketplace Portal.
- Relationships to other components can express both optional and required dependencies. During provisioning, a component can use the outputs of components upon which it is dependent.

Since the component is made of HP OO flows, each HP OO flow is represented by one component operation.

Import Wizard

To initiate the import in the Cloud Service Management Console, navigate to **Designs > Topology > Components** and click **Import**.



The Import Wizard appears.

Import Components [X]

☒ General

☐ Content Selection

☐ Parameter Mapping

Import Source:
HP Operations Orchestration [v]

Source Type:
Live instance [v]

Content Structure:
Standard Content [v]

Content Source

First you must specify the content source, as shown below. Pay special attention to the following:

- **Source Type** field. You can import either from a content pack file or from the live instance of an HP OO server configured in HP CSA. If you want to import from a content pack, you must first import that content pack to HP OO Central.
- **Content Structure** field. Select standard or custom content based on what content you have. Refer to [Standard versus Custom Content](#) above.

Import Components [X] **Standard Content**

☒ General

☒ Content Selection

☐ Parameter Mapping

Here you can select items for your component. Each item will be mapped to component operation. You can select multiple items from multiple folders.

🏠 / Library / Integrations / Hewlett-Packard / Cloud Servic...

📁 [..]

📁 Subflows

☒ Server (04.10.0000)

1 item selected

Previous Next Finish Cancel

Content Selection

Browse for the components you want to import. You can select multiple components from different locations if needed. Note that as a best practice the *standard content* is typically located in this path:

Library > Integrations > Hewlett-Packard > Cloud Service Automation > Provider > Component > Version

When importing *standard content*, you will select a leaf folder representing a specific version of a component. Consequently, you have to change the version folder name to import a new version of the content.

IMPORTANT: You cannot import a standard component that is the same version of one that already exists in HP CSA.

In contrast, *custom content* can be loaded from any path. When importing *custom content*, you are selecting HP OO flows. Together, all selected flows form one component. The flows are translated into the component operations. The new component name, description and associated provider type is entered manually during the last step of the wizard. The version of the custom component is automatically set to value **1** and cannot be changed. In addition, once the selected provider type is set, it cannot be changed.

The image displays two side-by-side screenshots of the 'Import Components' wizard for 'Custom Content'.

Left Screenshot (Content Selection):

- General:** Checked.
- Content Selection:** Selected (highlighted in blue).
- Parameter Mapping:** Unchecked.
- Details:** Unchecked.
- Instructions:** "Here you can select items for your component. Each item will be mapped to component operation. You can select multiple items from multiple folders."
- Path:** / Library / dev
- Items:** A list with a folder icon [..] and a checked item 'ExecuteRemoteCommand'.
- Status:** 1 item selected
- Buttons:** Previous, Next, Finish, Cancel

Right Screenshot (Details):

- General:** Checked.
- Content Selection:** Checked.
- Parameter Mapping:** Checked.
- Details:** Selected (highlighted in blue).
- Component Name:** * Tomcat
- Description:** Type description
- Resource Provider Type:** (none) (dropdown menu)
- Buttons:** Previous, Next, Finish, Cancel

Parameter Mapping

You can use predefined mapping or create a new mapping, as described below. Most of the components need input values for realization. These values come from user input via component properties, related components, or from the provider instance that is used for the realization.

The last group - **provider properties used by the component** - relates to the component import wizard step labeled **Parameter Mapping**. Typical provider properties used by the component include the provider endpoint and provider credentials.

Import Operations

General

Content Selection

Parameter Mapping

Use existing Parameter Mapping

Parameter Mapping:

Default VMware vCenter Parameter Mapping

| OO Flow Parameter Name | Provider Property | | |
|------------------------|-------------------|--|--|
| datacenterName | DATACENTERNAME | | |
| vCenterHost | HOST | | |
| vCenterPassword | PASSWORD | | |
| vCenterPort | PORT | | |
| vCenterProtocol | PROTOCOL | | |
| vCenterURI | URI | | |
| vCenterUsername | USERNAME | | |

Add Mapping Entry...

Previous Next Finish Cancel

To create a new parameter mapping for the purpose of component import, you need to know the properties of the imported component and identify which of these properties relate to the provider instance properties. Follow these steps:

- Get the information about component properties by examining the HP OO flows, HP OO documentation, or by trying the component import with the **No Mapping** option.
- Identify the properties related to the provider instance.
- Define the provider properties in the provider definition area under the Providers tile.
- Add mapping entries to map a component property to a provider property.

For example, the Default VMware vCenter Parameter Mapping maps provider properties to flow input properties as follows:

| Flow Input Parameter | Provider Property |
|----------------------|-------------------|
| datacenterName | DATACENTERNAME |
| vCenterHost | HOST |
| vCenterPassword | PASSWORD |
| vCenterPort | PORT |
| vCenterProtocol | PROTOCOL |
| vCenterURI | URI |
| vCenterUsername | USERNAME |

Note that provider properties come from the provider service access point information configured during resource provider configuration (for example, HOST, PASSWORD, or URI). Any additional properties (for

example, DATACENTERNAME) must be manually defined as a property on the resource provider using the **Properties** tab in the Resource Provider configuration area. In the case of *standard content* import, one or more components are created and all use the selected parameter mapping.

Post-Import Adjustments

Click **Finish** to create your new components. Depending on whether the component has been created using *standard* or *custom* content, a different level of configuration has been done automatically. Typically, a few more adjustments are required to make the component fully operational. For more information, go to [Fine-Tuning Imported Content](#).

HP OO Import Limitations

The *standard* content import works seamlessly. *Custom* content (which may not follow the best practices for content) can be imported as well in most cases; however, there are some known limitations.

HP OO Operations

Among the HP OO entities, there are flows and low-level operations. The HP OO operations are only building blocks that should be used as part of flows. Unlike to HP OO flows, HP OO operations cannot be imported as HP CSA components.

Single Flow Operations

An imported component operation invokes an HP OO flow when the component is provisioned as part of topology design provisioning. Each operation is linked to and can invoke just one HP OO flow.

Unique Lifecycle Operations

HP CSA defines the following lifecycle operations:

- Deploy
- Deploy Failure Handler
- Modify
- Unmodify
- Modify Failure Handler
- Undeploy
- Undeploy Failure Handler

A component can have a single lifecycle operation of each type; for example, a single *Deploy* operation. So, when an operation is set to *Deploy*, any other operation that was previously configured as *Deploy* will have its lifecycle operation unset.

Content Provided by Chef

Chef is an automation framework that deploys servers and applications to any physical, virtual, or cloud location. Chef operates with *cookbooks* consisting of *recipes*.

- Recipe – a script capable of installing something somewhere
- Cookbook – a folder containing a number of related recipes

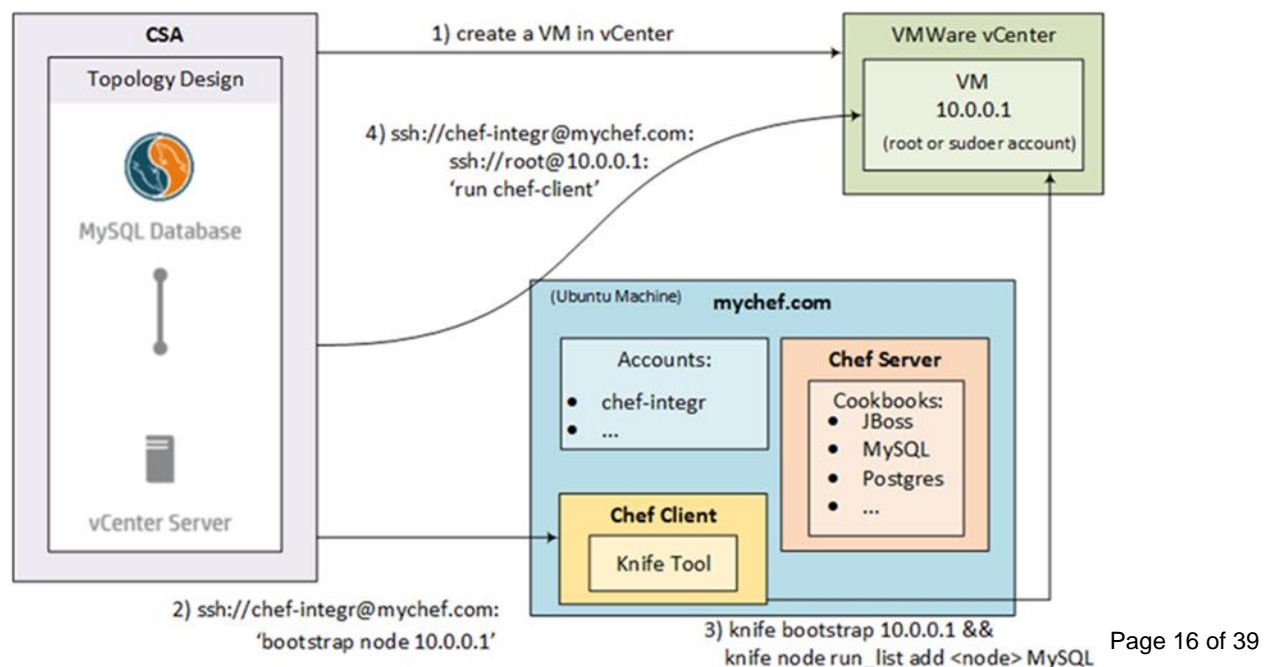
HP CSA leverages Chef cookbooks for installation of software on top of infrastructure provisioned by providers such as VMware vCenter or Amazon EC2. The Chef cookbooks are automatically encapsulated into HP CSA components, which can be used for topology design composition. HP CSA communicates with Chef using public REST-full APIs, collects information about each cookbook selected for import, and creates a component based on the information collected.

Once a component is ready, it can be used in a design. You define a design in the HP CSA Topology Designer. The Topology Designer allows service developers to associate Chef components with the infrastructure components on which the Chef components will be provisioned.

Figure 1 below shows the HP CSA integration with Chef.

1. HP CSA provisions the server infrastructure used to host the Chef software.
2. HP CSA connects to the Chef host machine via SSH and invokes the knife command located there to bootstrap the infrastructure machine (vCenter VM for example). A user account must exist on the Chef host with access to the Chef configuration files, keys, and executables.
3. On the Chef host machine, the `knife bootstrap` and `knife node run_list` commands are invoked to put the infrastructure machine under Chef management and to select cookbooks to install the requested software.
4. HP CSA connects to the Chef host machine via SSH and opens another connection to the server created in Step 1 to execute the `chef-client` runnable. The nested SSH call uses credentials of the infrastructure machine that have been input in the Topology design. Authentication via both password and private key is supported. The `chef-client` runs the installation script from the cookbook to install the software on the machine.

Figure 1: Chef Integration



Chef Setup

For Chef installation guidance, go to the Chef documentation (see [For More Information](#)). The section below contains useful hints for Chef installation and configuration within context of HP CSA.

IMPORTANT: The minimum supported Chef server version is 11.01. While HP CSA works well with the open-source version of Chef, the enterprise edition is **not** supported. For supported configurations, refer to the table below.

Supported Configurations

| Software | Supported | Notes |
|--|--|---|
| HP CSA Server OS | Windows 2008 Windows 2012 | See the HP CSA 4.2 Support Matrix |
| HP CSA Version | HP CSA 4.2 | See the HP CSA 4.2 Installation Guide |
| Chef server OS | Ubuntu 12.04 LTS | See Chef Server Installation for additional software requirements |
| Chef server software | Open-source Chef server, Version 11.01 | Hosted and Enterprise Chef not supported |
| Nodes provisioned by HP CSA used with Chef | Ubuntu and RHEL/CentOS 6.X | <code>ssh</code> is not installed on Ubuntu by default; make sure it is added. |

IMPORTANT: The system where the Chef server is installed as the management node should also have the Chef client installed. After the infrastructure is provisioned by HP CSA, the Chef client is used to bootstrap the provisioned system as shown in Step 2 of [Figure 1: Chef Integration](#) above.

Prerequisites

1. Install and configure an Ubuntu 12.04 server to host the Chef server and Chef client (Chef host).
2. Configure this system with a fully-qualified domain name and make sure that it is recognized by DNS.
3. Configure the system to use the same NTP time references and time zone settings as the HP CSA server.
4. Install additional utilities that are required for the HP CSA-Chef integration:


```
apt-get install sshpass
apt-get install netcat
```
5. Create a user account on the host machine for integration with HP CSA. This account will be used for SSH connections initiated by HP CSA. The user does NOT need to be **root** or **sudoer**; however, the account must have access to the knife tool, configuration file, and key. This user account will be referred to as **chef-integr** in the remainder of this document.
6. The Chef host machine requires network connection to machines provisioned by HP CSA so that software can be installed on those machines.

Chef server Installation

For Chef server installation on Ubuntu 12.04, follow these steps:

7. Get the open-source Chef installation here: <http://www.getchef.com/chef/install/>
8. Install the downloaded package using this command:

```
sudo dpkg -i chef-server*.deb
```

9. Run the Chef server configuration using this command:

```
sudo chef-server-ctl reconfigure
```

10. Check the status of Chef server with this command:

```
chef-server-ctl status
```

11. Use a web browser to connect to the Chef server:

```
https://<chefservername>
```

12. The initial credentials are as follows: **admin / p@ssw0rd1**

13. You will be prompted to change the password when you log in, as shown below.

IMPORTANT: DO NOT create a new user on the Chef server at this point in the installation.

The screenshot shows the Chef Server web interface. At the top, there's a navigation bar with 'Chef Server' and 'Environment: None'. Below it, a menu bar contains 'Environments', 'Search', 'Status', 'Roles', 'Nodes', 'Cookbooks', 'Databags', and 'Clients'. The main content area is titled 'Messages' and features a yellow banner with the text 'Please change the default password'. Below this, the 'Edit user: admin' section is active, showing a 'List' button and tabs for 'Create', 'Show', 'Edit', and 'Delete'. The 'Edit' tab is selected, displaying a 'Password' field with a masked password, a 'Password confirmation' field, and a 'Regenerate Private Key' checkbox. A 'Save User' button is at the bottom. The footer indicates 'Copyright © 2009-2014 Opscode'.

Chef client Installation

The Chef client needs to be installed on the same machine as the Chef server. HP CSA uses the knife tool (which is part of the client) to bootstrap provisioned nodes.

14. Go to <http://www.getchef.com/chef/install/> and download the client package

15. Install downloaded package with this command:

```
sudo dpkg -i chef_*.deb
```

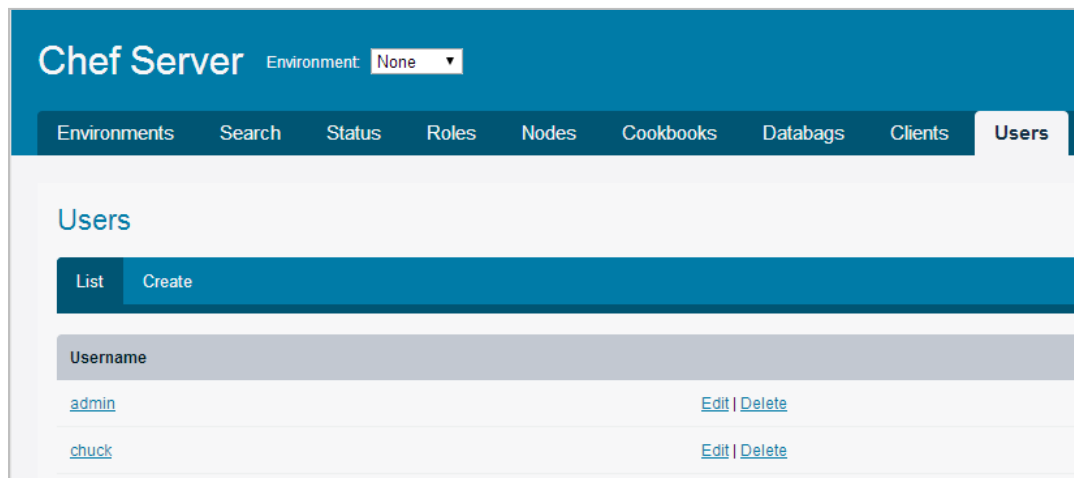
16. Configure knife by running this command:

```
sudo knife configure -i
```

You will be asked several questions. Most of them can be left with the default values. The most important values are the following:

- The Chef server URL must be FQDN like this: **https://mychefserver.com**

- The name and password for the new user needs to be created in the Chef server. We will refer to this user account as **chefuser** in the remainder of this document. A private key is generated for the new user. This username and the generated key are used by the knife tool to connect to the server. Keep these credentials safe. You will need to input them later in HP CSA.
- You can validate the new user in the Chef server's web interface, as shown below.



- The **configure** command creates a `.chef/knife.rb` file containing all configuration. Change the `.chef` owner to the system user that you want to use as the integration user in HP CSA. Grant the knife tool executable rights to this user.
- You may need to add a proxy configuration. Open the `.chef/knife.rb` file and add:


```
bootstrap_proxy 'http://your.proxy:8080'
```

Required User Accounts

This user configuration is required for the Chef – CSA integration:

17. CSA connects to the Chef server using SSH to invoke `knife bootstrap`. A system user (for example, **chef-integr**) is required for this call. The system user does not need to be **root** or **sudoer**. Access to the Chef configuration files, keys, and executables is required.
18. The knife tool connects to the Chef server using internal protocol. A Chef user account (for example, **chefuser**) together with the private key is required. This user account is created automatically during the knife configuration.
19. HP CSA connects to the Chef server public API to list cookbooks. The same Chef account (for example, **chefuser**) can be used.

Import Chef Cookbooks

To install common cookbooks, use the knife command. First download the cookbook from the Chef site, typically a `*.tar.gz` file.

```
knife cookbook site download apache2
```

To upload the cookbook, create subdirectory `cookbooks` and unzip downloaded file into it. Then upload the cookbook to the server using this command:

```
knife cookbook upload apache2 -o cookbooks
```

Some cookbooks depend on the other cookbooks, so you may encounter error messages from the knife command if dependent cookbooks have not already been uploaded. If you encounter error messages like

this, first download the missing cookbook from the Chef site and upload the cookbook using the knife command.

SSH setup

You may encounter problems with SSH IP validation if the server provisioning uses a limited set of IP addresses and client VMs are regularly created or deleted,. If the new VM uses an IP address that was previously in use on another VM, SSH can prohibit the access because of an identity conflict. You can solve these problems by disabling the validation. In file `/etc/ssh/ssh_config` set these options:

```
CheckHostIP no
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
```

VM Template Working with Chef

Chef cookbooks will be installed on Server components that are provisioned by HP CSA using VM templates. This section explains the process of creating and configuring a VMware vCenter template for use in a Chef environment. If you will be using Amazon for server provisioning, you must rely on templates provided by Amazon (however, configuration often works seamlessly in an Amazon environment).

A VM template is a standard image of the OS with some basic networking configuration and required tools installed. Typical examples of additional configuration are as follows:

- proxy settings
- yum/apt-get installation and configuration
- wget installation and configuration

Currently, only Linux templates are supported for the virtual machines used with Chef components. Most Chef cookbooks are written to work with `apt-get` or `yum`, so HP recommends using Ubuntu and CentOS distributions. Theoretically, any Linux template should work if it provides the required tools and satisfies cookbook requirements.

Steps to Create VM Template

1. First create a standard VM machine in the vCenter client wizard.
2. Install the OS (CentOS 6.4 in this example, which is similar for Ubuntu or RHEL).
3. Set up the proxy.
 - If the machine is behind a proxy, you must configure the proxy in the template. First add these lines into the `.bashrc` file in the root home directory:

```
export ftp_proxy=http://your.proxy:8080/
export http_proxy=http:// your.proxy:8080/
export https_proxy=http:// your.proxy:8080/
export no_proxy=localhost,127.0.0.1
```

- You also need to specify proxy for yum. In the file `/etc/yum.conf` add this line:

```
proxy=http://your.proxy:8080
```

4. Install perl and wget:

```
yum install perl
yum install wget
```

5. Setup the proxy for wget in the file `/etc/wgetrc`:

```
https_proxy = http://your.proxy:8080/
http_proxy = http://your.proxy:8080/
ftp_proxy = http://your.proxy:8080/
```

6. Disable or configure the firewall:

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
chkconfig iptables off
```

7. Shutdown the VM, then right click the VM in vCenter and select **Template > Convert to Template**.

Customization Specification

The customization specification is created in the vCenter client, specifying the network setup and the hostname convention of the provisioned VMs. The custom specification also allows the return of the IP address and name of the provisioned VMs to HP CSA. This configuration ensures that machines you create can be accessed from HP CSA and from Chef servers. You can find configuration details for the customization specification in the VMware documentation (see [For More Information](#)).

The Chef Provider in HP CSA

The Chef server must be registered as a resource provider in HP CSA. Server endpoint, credentials, and other properties are required.

Edit Resource Provider

Provider Type
Chef

Display Name *
chef

Description

User ID *
chef-integr

Password *

Service Access Point *
https://mychef.com

Image
Change Image
Recommended dimension of 256x256. Maximum file size of 1MB.

Default Settings
Enabled

Save Cancel

- **User ID** is the system user account existing on the Chef server machine (for example, **chef-integr**). This user account connects to the machine via SSH.
- **Password** secures the user account.

- **Service Access Point** is the Chef server endpoint (default port is 443).

`https://<hostname>:[port]`

Besides the standard access information, two more properties are required. They are created under the resource provider **Properties** tab.

- `chefClient` is the client/user name configured on the Chef server. This name is used for access to the Chef API.
- `chefClientKey` is a private key securing the client above.

The client name and key values are collected when a new user is created on the server. You can use the user created during Chef client configuration or manually create a new user using the Chef server UI.

Import Cookbooks as Components

Chef operates with cookbooks consisting of recipes. The recipes are Ruby scripts automating the installation and configuration of software pieces such as databases or application servers. During the import, Chef cookbooks are encapsulated into HP CSA components, which can be used for a topology design composition.

Imported Chef components are not bound to the original Chef instance in any way. So a design containing Chef components can be provisioned using any other compatible Chef server, as long as the required cookbooks are in place.

HP CSA communicates with the Chef server using public REST-full APIs, collects information about each cookbook selected for import, and creates a component based on the information collected. This information is presented to the user as the Component Import Wizard. The Component Import Wizard is located under **Designs > Topology > Components** in the Cloud Service Management Console.

Content Source

Chef cookbooks are loaded from a running Chef server. You configure Chef server instances as resource providers in HP CSA. If you configure multiple Chef resource providers, one must be selected.

Import Components

☒ General

☐ Content Selection

Import Source:
Chef

Source Type:
Live instance

Provider Instance:
Chef Server in Amazon

Previous Next Finish Cancel

Content Selection

You can select one or more cookbooks for import. A cookbook version is displayed in parentheses after the name. The cookbook version is used as part of the new component name, so that multiple cookbook versions can be imported as different components. Note that the cookbook version is not translated into the component version. The component version is automatically set to **version 1** and auto-incremented when the cookbook is imported repeatedly.

Import Components

General

☒ Content Selection

Here you can select Chef cookbooks which you want to import as new CSA components. You can select multiple cookbooks each creating one component.

| | |
|-------------------------------------|---------------------------|
| <input type="checkbox"/> | git (2.9.0) |
| <input type="checkbox"/> | ipaddr_extensions (0.1.0) |
| <input checked="" type="checkbox"/> | java (1.19.2) |
| <input checked="" type="checkbox"/> | jboss (0.0.3) |
| <input checked="" type="checkbox"/> | mysql (4.0.20) |
| <input type="checkbox"/> | openssl (1.1.0) |
| <input type="checkbox"/> | postgres (1.0.1) |
| <input type="checkbox"/> | rbac (1.0.1) |

3 items selected

Previous Next Finish Cancel

Recipes in a Cookbook

Chef cookbooks consist of one or more recipes. When a new component is created from a cookbook, HP CSA always uses the *default* recipe as the component implementation. If another recipe is required, some manual work must be done.

To change the recipe executed during the component deployment, edit the component **Deploy** operation and look for the `chefRecipeName`. The value can be any valid run-list expression `cookbook[@version][:recipe]`.

Edit Operation

General

Interface

☒ Implementation

Action:
Deploy

Parameter Mapping:

| Parameter | Value | Source |
|------------------|----------------------|------------------------------|
| chefHostPassword | | Provider Property |
| chefHostUsername | USERNAME | Provider Property |
| chefRecipeName | mysql@4.0.20 | Constant Value |
| chefTimeout | chefTimeout | Provider Property |
| nodeIPAddress | hostedOn > ipAddress | Relationship Target Property |
| nodeIpAddress | nodeIpAddress | |

Previous Next Finish Cancel

Example values of `chefRecipeName` parameter are as follows:

- `mysql`
- `mysql::client`
- `mysql@4.0.20`
- `mysql@4.0.20::client`

Different recipes from one cookbook or multiple versions of one cookbook can coexist in HP CSA as separate components. It is useful to clone the component first by clicking **Save As** and then changing the recipe values for individual copies.

HP CSA uses Chef metadata to determine what is inside a Chef cookbook and how its recipes and attributes (inputs) should be used. If a recipe or attribute (input) is not properly defined in the cookbook's *metadata.rb* file, it is not recognized by HP CSA. The cookbook metadata file is described in Chef documentation: http://docs.opscode.com/essentials_cookbook_metadata.html

Modification of Chef Components

A Chef component is modifiable during the Modify Subscription operation, provided that the Chef component's cookbook is *idempotent* (written in a manner that the same cookbook can be applied more than once). When a Chef component is embraced by importing a Chef cookbook, HP CSA assigns the default recipe to the Modify lifecycle phase as well as to the Deploying phase, so effectively the same cookbook is used for both initial deployment and subsequent modification requests.

Testing and Debugging Chef Components

HP CSA wraps Chef cookbooks as components. During provisioning, HP CSA triggers the execution only. The cookbooks are executed if Chef alone was involved. If something goes wrong, the user is informed via HP CSA about problems. However, the information is not as detailed as it can be when working with Chef directly. To get more details, connect to the provisioned machine as the node managed by Chef and invoke *chef-client* directly.

Typical problems:

- Networking problems. Connections are required between:
 - HP CSA (and HP OO) and the Chef host
 - HP CSA and the infrastructure provider used for machine provisioning
 - HP CSA (and HP OO) and the provisioned machine
 - Chef host and the provisioned machine
- The cookbook does not work well on the operating system installed on the provisioned machine; for example, the cookbook works for a specific Linux distribution only.

Fine-Tuning the Imported Content

Once the content is imported, components are created. The components usually need some additional adjustments, such as relationship definition, lifecycle configuration for operations, or operation parameter mapping. The amount of additional work required depends on the nature of the imported content and user requirements.

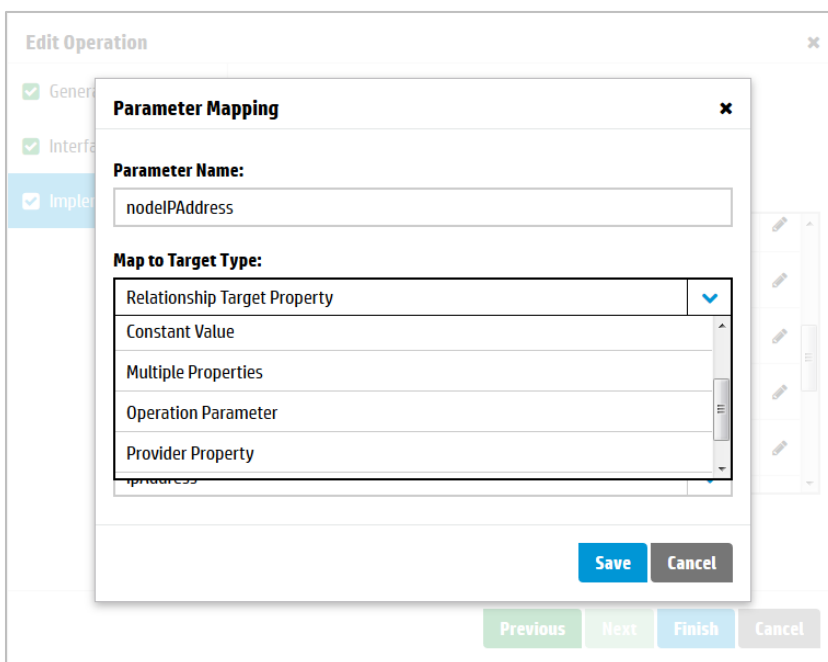
Component Overview

The imported content is represented as components in HP CSA. Components are parameterized with input properties, have relationships to other components, go through a lifecycle during operation execution, which can generate values for output properties and optionally provide additional operations, called *public actions*, which can be requested by Subscribers.

Before you start modifying components, note the following:

- The Server capability is read-only and cannot be changed.
- The HP Helion OpenStack components are read-only and cannot be changed.
- Other components can be changed; however, when a component is already used in a design, changes are restricted to the following:
 - The component name, description, and icon can be changed.
 - The component can be tagged.
 - Non-required properties can be added.
 - Non-required relationships can be added.
 - Custom non-lifecycle operations (public actions) can be added.
 - Other changes related to required properties, relationships, and lifecycle actions are **prohibited** once the component is used in a design.

The most important part of a component is operations, so use caution when configuring operation parameter mapping.



There are several ways to get a value for an operation input parameter.

Not Mapped – The operation parameter automatically gets no value. This is an invalid state for a lifecycle operation (for example, *Deploy*, *Undeploy*). All parameters must be mapped. However, in the case of a public action, the *Not Mapped* parameter is a valid option and results in a value prompt during execution.

NOTE: The known defect that breaks the value prompt will be fixed in the next patch.

- *Component Property* – The operation parameter is filled using the component property value (the most straightforward way).
- *Constant Value* – The operation parameter value is a constant.
- *Multiple Properties* – A value for the operation parameter is combined from multiple places. This is a kind of recursion, so another level of mapping resides here.
- *Operation Parameter* – This option should not be used as it has not been fully implemented.
- *Provider Property* – The operation parameter uses a property defined on the provider instance used for provisioning.
- *Relationship Target Property* – A property defined on another component connected via a relationship is used for the operation parameter. In case the selected relationship is not in place, the property with the same name (if present on the component) is used. The behavior is the same as Component Property mapping. A fallback mechanism is used to map this parameter to a property with the same name on the component itself, if that property is present in the component.

Server Capability

Topology components do not form a hierarchy (with the exception of HP Helion OpenStack components). Useful aspects of inheritance are covered by the concept of *capabilities*. A capability can be considered a tag indicating what a component is good for or capable of; however, it is more than just a tag. A capability can have properties and relationships, making it look more like an abstract component.

HP CSA provides some out-of-the-box capabilities (for example, Server, Platform, Database Server, Application Server, and Web Server). Each of these capabilities has a different set of properties and relationships. For example, components with Server capability have server-related properties: `ipAddress`, `hostname`, `username`, `password`, `privateKey`, and `instanceId`. Moreover, all Chef components have a relationship called `hostedOn` that is automatically created after import. The `hostedOn` relationship targets the Server, so it can be used to get the Server properties for the component deployment.

Note that when a Server component is used at design time, you must decide which authentication method should be used by provisioning. If you use `username + password`, you must leave the `privateKey` field empty; otherwise, `privateKey` is used.

Lifecycle Operations

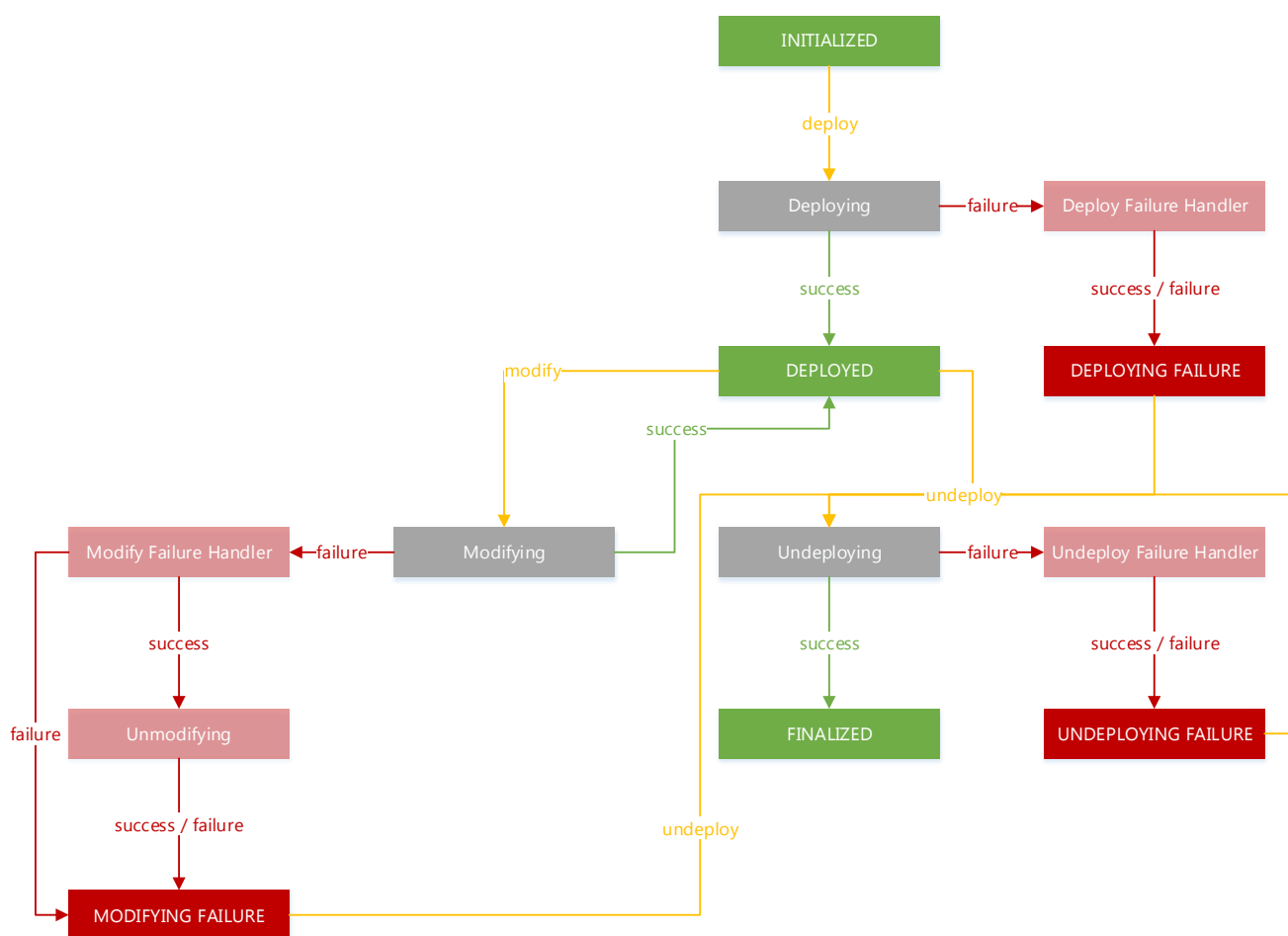
Components in HP CSA go through lifecycle stages as they are provisioned. The lifecycle of topology components is simpler than the lifecycle of sequenced components. The information below displays in both the HP CSA Operations Console (Management Console) and in the Marketplace Portal.

- Green boxes are stable success states.
- Red boxes are stable failure states.
- The faded gray and pink boxes are execution states. Transition to the next state triggers once the execution is finished.
- Yellow transitions from a stable state trigger when a component operation is explicitly invoked.
- Other transitions are automatically triggered.

The lifecycle operations that may be defined on a component are Deploy, Undeploy, Modify, Unmodify, Deploy Failure Handler, Undeploy Failure Handler, and Modify Failure Handler.

A Deploy operation is triggered when a new request is submitted. An Undeploy operation is triggered when an existing subscription is canceled. A failure in deploying or undeploying automatically invokes the corresponding failure handlers.

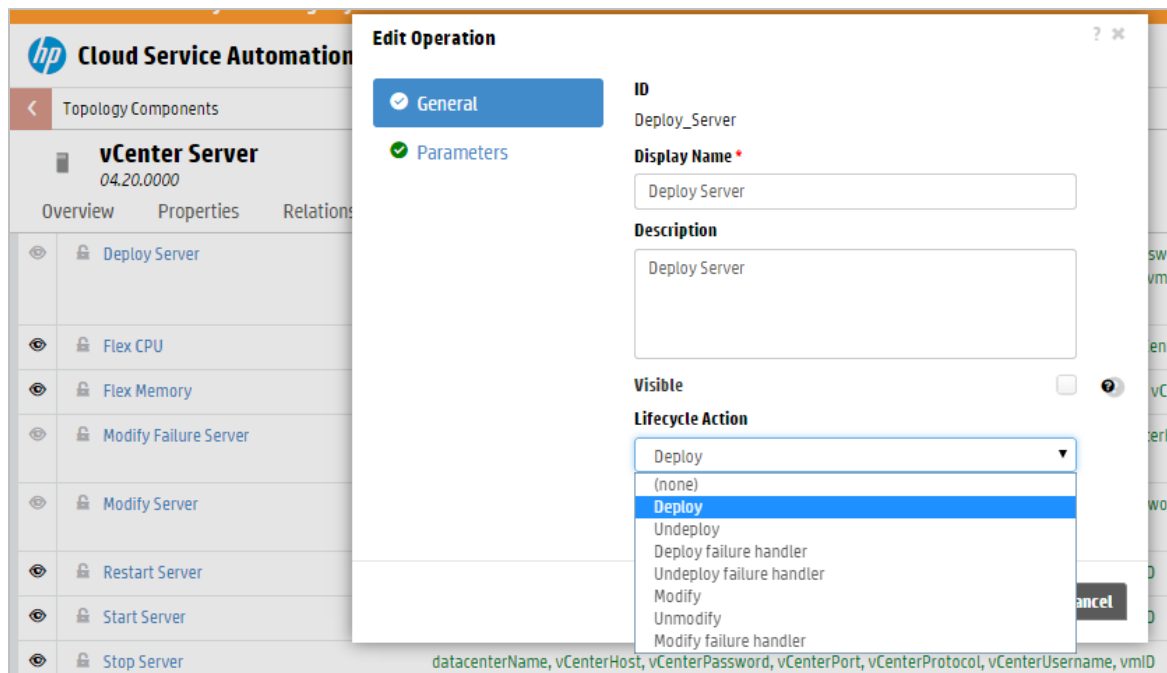
A previously deployed subscription may be modified. When the modification operation is successful the component moves back to the Deployed state. If a failure occurs during modification, HP CSA attempts to rollback all the operations executed so far in the order they occurred. This rollback reverts the components to the state prior to the modification attempt. A failure at the time of modification invokes the Modify Failure Handler, which operates on the component whose modify action failed. This in turn triggers the Unmodify operation on the component that has been successfully modified to achieve a clean rollback of modification transition. If the “un-modification” was successful, the component’s properties are rolled back. However, if the un-modification was not successful, the component’s properties are retained and the Consumer can resubmit the last modification. It is likely that resubmitting the last modification will result in failure similar to previous one, so the Operator may have to intervene to resolve the situation.



NOTE: During modification of a service instance, the modify action is invoked on **all** the components in the service instance regardless of whether or not the modification request attempted to change component property values. However, the modify operation is passed with the previous value and new value for the modifiable properties, so a Component Developer can decide whether any modification should be performed by comparing the two values (previous and new) and request a **noop** response, if no change was made to the component. It is a good practice to check whether the underlying resource currently exists before trying to modify or un-modify that resource.

To define a lifecycle operation, edit the component, select an operation, and select the appropriate value from the dropdown menu.

NOTE: Lifecycle operations are automatically recognized if the component has been created using a Chef cookbook or standard HP OO content.



Custom Operations

Besides the operations automatically created for you during content import, you can define additional custom operations. The **Import** button on the Operations tab initiates a wizard similar to the wizard used for component creation (see

HP OO Content Import). Just select an HP OO flow that you want to use as the implementation of the new Operation.

Component Properties

The newly created component comes with a set of input and output properties that match the HP OO flow input and output. That way the component is a wrapper around the HP OO flow or the Chef cookbook used for the deployment. If you want the component to be more than that, you need to understand component properties.

There are several typical groups of properties on a component.

- Properties related to the resource provider, such as service endpoint and credentials for Amazon EC2.
- Properties related to other components, such as the IP address of the server used for installation
- Properties parameterizing the component.

Properties from the first group of resource provider-related properties are not shown on the component if a proper parameter mapping has been used during the content import (see [Parameter Mapping](#)). Note that for Chef, the provider parameters are handled automatically, specifying mapping.

NOTE: In the case where components have a defined relationship among themselves, the following applies:

When the value of an input parameter of an action on a component is captured from the property on another component, it is not necessary to define a property for that input parameter in the component itself. See [Relationships To Other Components](#) below.

The last group is properties that belong to the component and must be specified by a user.

A property can be configured so that it is required or optional, visible or hidden in the Designer, modifiable, or has a default value. Property types can be selected and string properties can be set as *confidential*. Confidential properties are rendered so the value is not visible in plain text.

NOTE: All values are passed to the HP OO master-flow during provisioning. Confidential values are obfuscated in HP OO. Out-of-the-box components are an exception in that they have the password obfuscation already implemented.

Edit Property

Display Name: * nodeIpAddress

Type: * String

Description: Type Description

Default Value: Type a default string value for property

Can be Modifiable: Yes (checked by default)

☐ Required Property

☐ Visible Property

Modifiable property can be changed at provision time.

Ok Cancel

Relationships to Other Components

A component usually works with other components in deployments that are more complicated than the deployment of a single server. Topology designs capture the component dependencies and connections using the concept of *relationships*.

A relationship has several functions:

- Denotes dependency between two components in a topology design.
- Specifies the order of component realization during provisioning, which is driven by dependencies represented by the relationship.
- Defines property values that can be passed from one component to another along the relationship. Relationships are oriented from a source component to a target component. The orientation is important because it marks a dependency of the source component on the target component. If Component A has a relationship to Component B with the arrow pointing to Component B, it implies that Component A depends on Component B and will be realized after Component B is realized.

Passing Values between Components

The common example of a relationship is a piece of software installed on a server. *MySQL* component realized by a Chef cookbook has a relationship *hostedOn* targeting *Server* capability. During the provisioning, the *MySQL* reads the server IP address using the *hostedOn* relationship.



At provisioning time, HP CSA resolves dependencies between components and creates the server first and the database second. When creating the database, HP CSA needs to know the server IP address so that it can run the Chef cookbook for MySQL installation on it. The provisioning is done by the Deploy lifecycle operation. The Deploy operation has *nodeIpAddress* as one of the inputs. The value is captured using the relationship from the *vCenter Server* output property, which has been already provisioned. The following formula describes how the IP address value used for the MySQL deployment is retrieved from the target of the *hostedOn* relationship which is the vCenter Server.

```
MySQL.Deploy.nodeIpAddress = MySQL.hostedOn.ipAddress
```

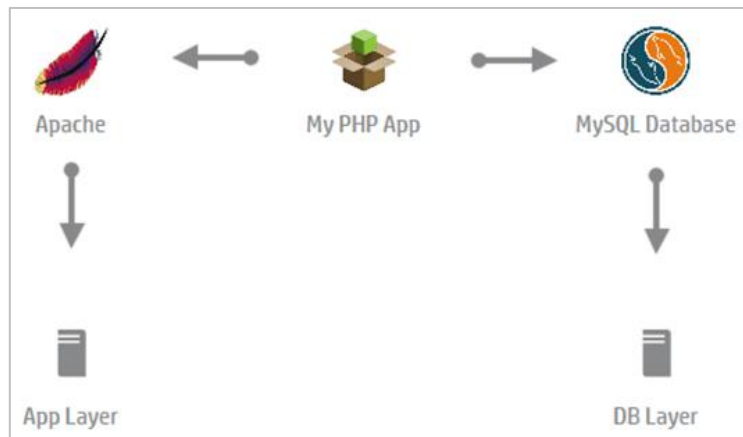
This value passed is configured in component editor. You can edit an operation and configure the parameter mapping in the Implementation section.

The screenshot shows the 'Edit Operation' dialog box with the 'Implementation' tab selected. A 'Parameter Mapping' sub-dialog is open, displaying the following configuration:

- Parameter Name:** nodeIPAddress
- Map to Target Type:** Relationship Target Property
- Relationship:** hostedOn (hostedOn_c56fbb25-36a2-40fb-886d-085b99cfb739)
- Property:** ipAddress

At the bottom of the 'Parameter Mapping' dialog are 'Save' and 'Cancel' buttons. At the bottom of the 'Edit Operation' dialog are 'Previous', 'Next', 'Finish', and 'Cancel' buttons.

There are scenarios where the value must be passed along with multiple relationships; for example, the two-tier LAMP design below. *My PHP App* needs to know the `ipAddress` of the *App Layer* server, so that *My PHP App* can be installed on *App Layer*. *My PHP App* also needs to know the `ipAddress` of the *DB Layer* server, so that the application can connect to MySQL. In both cases, the `ipAddress` flows along two relationships.



Since *My PHP App* is a Chef component originally, the component has a direct *hostedOn* relationship defined targeting *Server*. That relationship could be used for the `ipAddress` mapping. This scenario assumes manual replacement of the *hostedOn* relationship with the *webserver* relationship targeting *Apache*. Even keeping the original relationship could solve only the first half of the problem: the hosting server IP. It cannot solve the second half: the database server IP.

A Chef component accepts the *Server* properties using the *hostedOn* relationship. After the component is provisioned, output property values are set including (original input) values for `nodeName` and `nodeIpAddress` properties. When using Chef components, the `nodeName` and `nodeIpAddress` properties can be passed along a chain of multiple related components where each consumes output of the preceding one. This works only for Chef components, not for HP OO components. In symbolic notation this configuration looks like this:

```
MyPhpApp.dbIpAddress = MyPhpApp.database.nodeIpAddress = MySqlDb.hostedOn.ipAddress
```

This approach cannot be used with HP OO components. With HP OO components, a value has to be referenced using multiple relationships or *multi-hop* mapping:

```
MyPhpApp.databaseIpAddress = MyPhpApp.database.hostedOn.ipAddress
```

There is no way to configure this type of parameter mapping using the UI editor, as only single relationships are supported. However, there is a workaround enabling *multi-hop*. The public REST-full API for topology component management can be used to configure multi-relationship mapping. Once configured, the mapping works well on the backend.

The example below shows part of the PUT request JSON body used for a component update, giving the value flows along with the *webServer* and *hostedOn* relationships. The red text shows notation for the *multi-hop* mapping:

```

...
"mapping" : {
  "input" : {
    ...
    "nodeIPAddress" : "@PROPERTY:{urn\\:x-
hp\\:2013\\:software\\:ccue\\:designer\\:topology-metamodel\\:chef}webServer7638f7c8-a4c1-
478d-aaad-3844def94428.hostedOn_410ebb81-93a0-4c2e-b9c7-1f63b56b5ecf.ipAddress",
    ...
  },
  "output" : {
    ...
  }
},
...

```

Testing and Debugging

To test a component, you must use it in a design and test the entire design. HP CSA provides a Test Run feature for topology designs. Test Run results include events details together with links to the OO master-flow execution log.

Refer to the Chef and OO related hints and limitations in chapters [HP OO Import Limitations](#) and [Error! Reference source not found.](#) For a deeper level of debugging, you can load master flows into OO Studio that are generated when a Test run is performed in HP CSA. These master flows are for provisioning a service, modifying a service, or de-provisioning a service. The content jar file for the generated master flows is saved in the folder `<CSA_INSTALLDIR>\jboss-as\standalone\tmp\e2e-flows`

Modification Scenarios

With a typical modification, you can expect that Modify operations for each of the components will run successfully to completion. The picture below shows a successful execution of the Modify operation for two components.

A successful Modify operation should return a response of **success**.

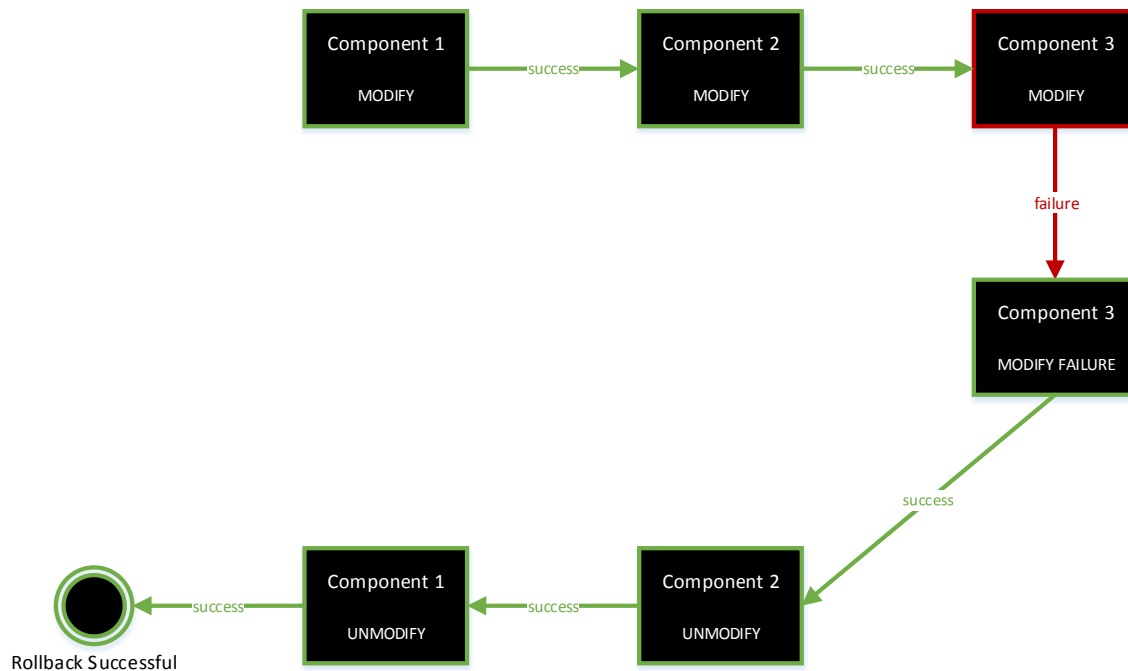


If the Modify operation for a particular component fails, its corresponding Modify Failure Handler is invoked. The Modify Failure Handler provides an opportunity for cleanup. The `modifyReturnValue` output from the Modify operation feeds into the `modifyFailureValue` input of the Modify Failure Handler. Information must be communicated from the Modify operation to its corresponding Modify Failure Handler to provide additional contextual information about where the failure occurred. The nature of the failure is *modify* action so that the Modify Failure Handler can do intelligent cleanup.

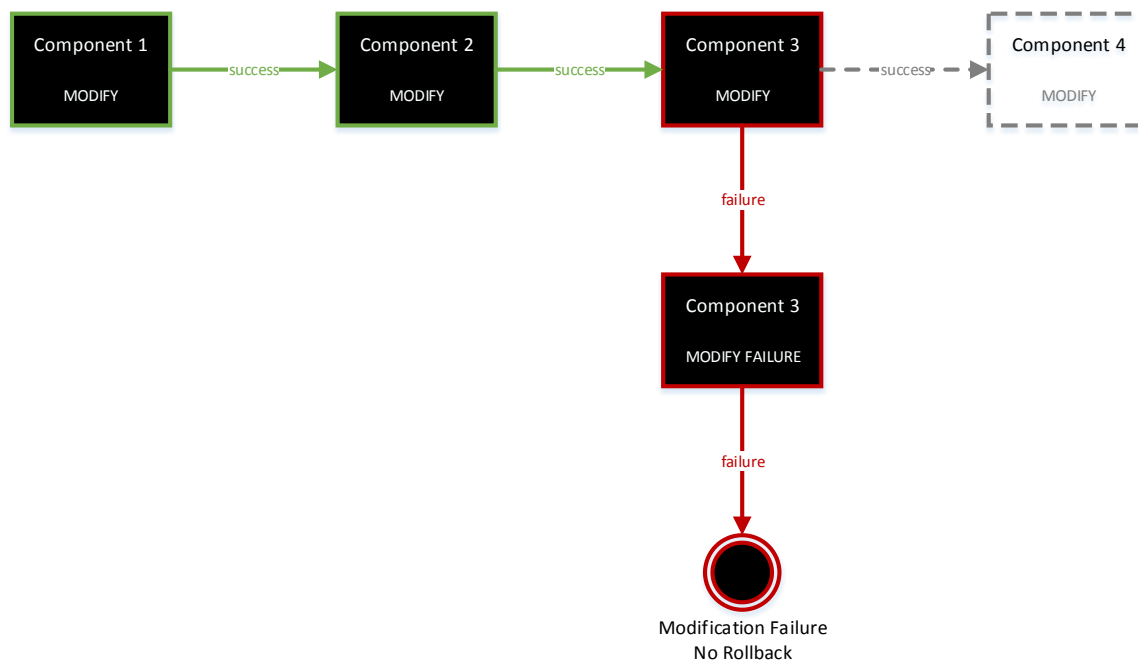
On the successful completion of the Modify Failure Handler, a rollback of previously deployed components is initiated by calling the Unmodify handler. When all previously modified components are successfully unmodified, these components move back to the Deployed state, while the component that failed (Component 3 in the picture below) is set to Modifying Failure.

Additionally, the options selected when *Modify Subscription* was initiated are reverted if the rollback is successful. Subscribers are free to make different selections while resubmitting their previously failed modification request.

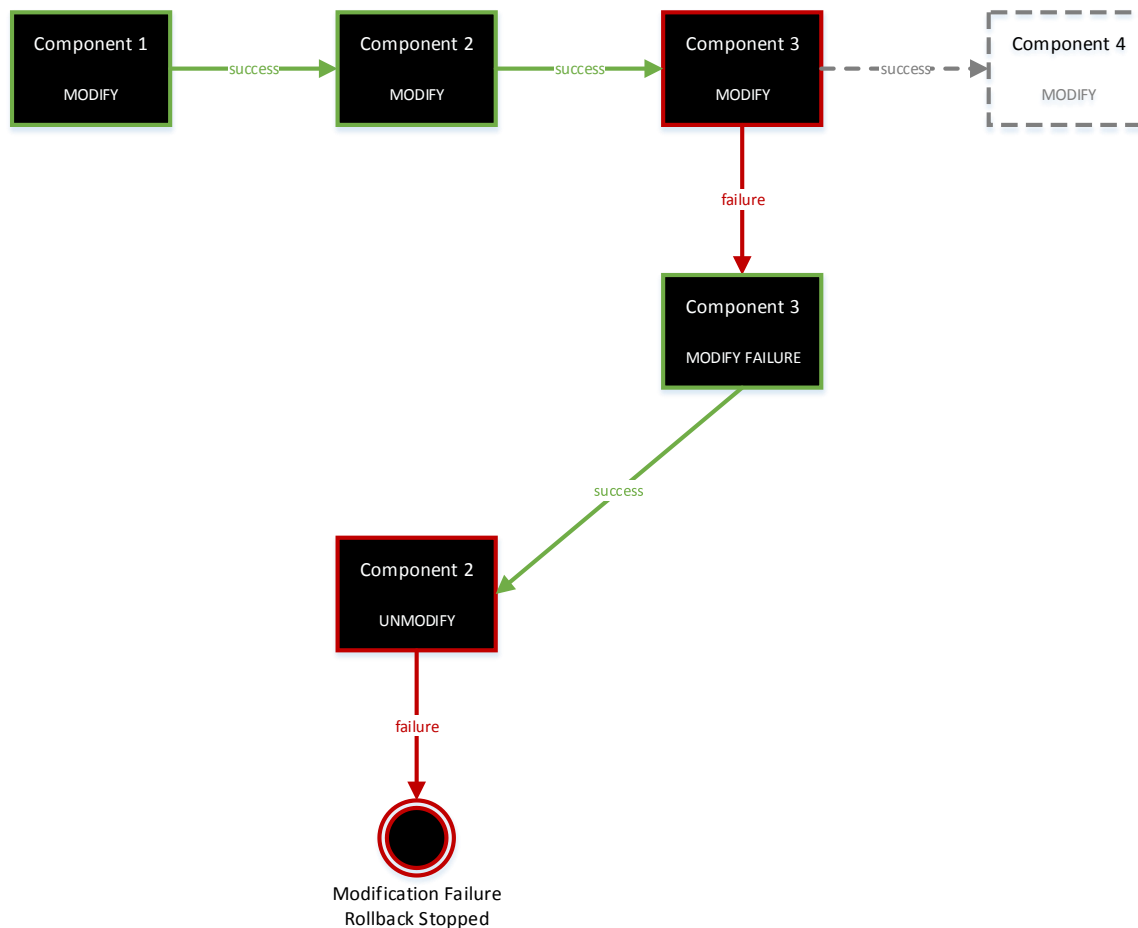
The Modify Failure Handler operation must always set the value of response parameter to failure. In the example below, the **failure** response indicates that the modification failed for Component 3. A successful Unmodify operation returns a response of **noop**, indicating that the rollback for that component was successful. A failed Unmodify operation returns a response of **failure**.



Failures may occur while executing the Modify Failure operation or Unmodify operations, as shown in the next two examples.



A rollback is either not initiated or stopped as soon as a Modify Failure Handler or an Unmodify operation fails. In both these scenarios, the Subscriber may not change options that were previously selected for a subsequent resubmission. It is likely that the Operator may have to intervene.

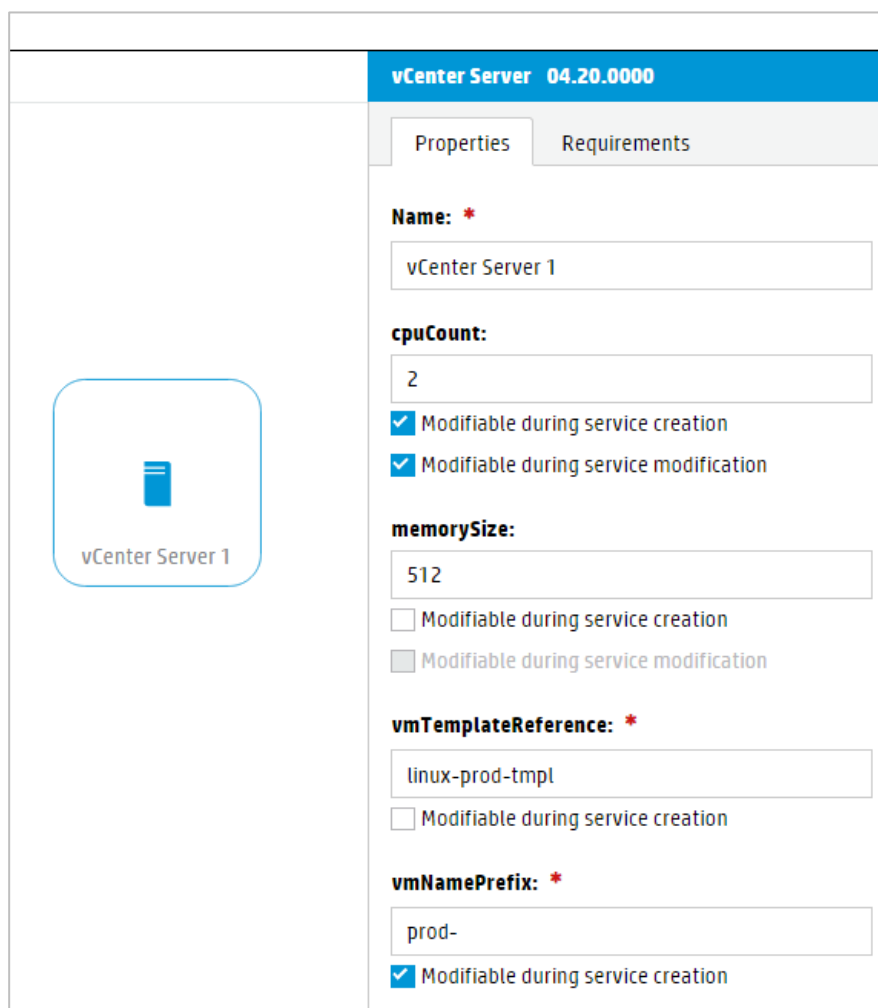


Configuring Modifiable Properties

Properties of a topology-based service design must explicitly be marked as *modifiable*. Designers may choose to mark properties as *modifiable* at the time of service creation, or both at the time of service creation and service modification.

Only properties defined with their `prev_` counterparts may be marked as *modifiable* during service modification. In the example below, `cpuCount` and `memorySize` are properties the Designer can mark as modifiable; the Designer has chosen only to mark `cpuCount`. Notice that the other properties, such as `vmTemplateReference`, `vmNamePrefix`, cannot be marked as modifiable because they did not have a corresponding `prev_` input parameter in the underlying Modify flow.

Properties on various profiles may also be marked as modifiable during service creation and modification.



The screenshot displays the configuration interface for a vCenter Server. On the left, there is a visual representation of the server with the label "vCenter Server 1". The main panel on the right is titled "vCenter Server 04.20.0000" and contains two tabs: "Properties" and "Requirements". The "Properties" tab is active, showing several configuration fields and checkboxes.

| Property | Value | Modifiable during service creation | Modifiable during service modification |
|------------------------|------------------|-------------------------------------|--|
| Name: * | vCenter Server 1 | | |
| cpuCount: | 2 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| memorySize: | 512 | <input type="checkbox"/> | <input type="checkbox"/> |
| vmTemplateReference: * | linux-prod-tmpl | <input type="checkbox"/> | |
| vmNamePrefix: * | prod- | <input checked="" type="checkbox"/> | |

Terminology

The following terms are used in this document.

Capability

A capability is a special type of component that supports properties and relationships (but does not support operations or characteristics). When a concrete component supports a capability, the concrete component inherits the capability's relationships and must provide property mappings from the concrete component properties to the capability properties. Capabilities can be the target of relationships configured on a component. Capabilities can be included in a design, but for such a design to be successfully provisioned, another design must exist that contains a concrete component supporting the capability.

Chef cookbook

A Chef cookbook is a collection of Chef recipes grouped together.

Chef recipe

A Chef recipe describes how Chef manages a server application and/or utility (such as MySQL) and how it is to be configured.

Component (in Topology Designer)

A component represents one service design element required to realize a service subscription. HP Cloud Service Automation provides a number of out-of-the-box components you can use for creating topology designs.

Component Properties (in Topology Designer)

Component properties provide a base set of attributes that can be used and edited when creating components in a service design. They represent configuration settings to be applied to the component during service design provisioning. The value defined for a component property is the default value exposed in the service design.

Component Types

A base set of attributes that can be used and edited when creating service components in a service design.

Content (standard versus custom)

Content can be imported from HP Operations Orchestration either in a standard form (known as Standard Content) or a custom form (known as Custom Content). Standard content must follow specific directory and naming conventions to infer the provider type, component name, and version of a created component (with the flow and subflows located under a `Provider Type Name / Component Name / Version` directory). Custom content requires the user to specify the provider type and component name at import time.

HP Cloud Service Automation

HP Cloud Service Automation (HP CSA) is a unique platform that orchestrates the deployment of compute and infrastructure resources and of complex multi-tier application architectures. HP CSA integrates and leverages the strengths of several HP datacenter management and automation products, adding resource management, service offering design, and a customer portal to create a comprehensive service automation solution. The HP CSA subscription, service design and resource utilization capabilities address three key challenges:

The HP CSA Marketplace Portal provides a customer interface for requesting new cloud services and for monitoring and managing existing services, with subscription pricing to meet your business requirements.

The HP CSA graphical service design and content portability tools simplify developing, leveraging, and sharing an array of service offerings that can be tailored to your customers' needs.

The HP CSA lifecycle framework and resource utilization features ease the complexity of mapping your cloud fulfillment infrastructure into reusable, automated resource offerings for on-time and on-budget delivery.

HP Cloud Service Management Console

Software that provides an HP CSA design and administration interface. The Cloud Service Management Console supports the following user roles: Administrator, Consumer Service Administrator, Resource Supply Manager, Service Designer, and Service Operations Manager.

HP Marketplace Portal

Software that delivers cloud services to subscribers (customers) by providing one or more service catalogs per organization. The Marketplace Portal is integrated into and shipped with HP CSA. The Marketplace Portal supports the following user roles: Consumer Organization Administrator, Consumer Organization Administrator, and Service Consumer.

HP Operations Orchestration

HP Operations Orchestration (HP OO) is a software product that coordinates communication between integrated products and managed devices. Customized HP OO flows are essential to implementing the HP CSA service lifecycle.

HP Operations Orchestration Flow

A run-book automation workflow composed of operations, subflows, and integrations which implement a discrete action. Flows are synchronized with HP CSA, and presented as actions, which can be directly attached to components. HP Operations Orchestration flows are created, modified, and saved using HP Operations Orchestration Studio. HP CSA includes a set of sample HP Operation Orchestration flows used by HP CSA's sample service designs.

HP Server Automation

HP Server Automation is full-scale virtual server management automation software that delivers security, provisioning, policy management, deployment, and server compliance.

Lifecycle (in Topology Designer)

Each Topology component goes through a lifecycle transition: create, modify, and finally destroy (when not needed by the service anymore). These transitions throughout the life of the component are called the *lifecycle phase*.

Lifecycle Operations (in Topology Designer)

A topology component goes through various lifecycle phases: for example, Deploy, Modify, Un-deploy. Operations executed on these Lifecycle Phases are called *Lifecycle Operations*.

Parameter Mapping (in HP CSA)

Each Topology component defines a set of actions and each action typically declares a set of input parameters. A parameter mapping is defined to specify how the value should be passed for those input parameters when the action is executed. For example, value of an input parameter may be mapped to a component's property by creating a parameter mapping. In that case the value of the property is passed as the value of the input parameter when the action is executed.

Properties (modifiable properties)

Properties provide a base set of attributes that can be used and edited when creating components in a service design. They represent configuration settings to be applied to the component during service design provisioning. The value defined for a component property is the default value exposed in the service design.

Relationship

Relationships in topology designs define dependencies between components and also impact how a design is provisioned. For example, imported Chef components require a Server in order to be provisioned. Therefore, all imported Chef components are created with an Outgoing relationship to the Server capability, ensuring that a Server is provisioned before the Chef component.

Resource Provider

A management platform that provides either Infrastructure-as-a-Service (IaaS) or Software-as-a-Service (SaaS) to the cloud. For example, a provider of HP Matrix Operating Environment services provisions infrastructure and basic applications, while a provider of HP SiteScope services monitors applications.

Topology Design

Topology designs specify components, relationships, and properties. In contrast to sequenced designs, which more explicitly define the provisioning order and the sequence of actions that will run, topology designs are declarative in nature and do not include explicit actions or sequencing. The provisioning sequence is inferred by the relationships that exist between components in a topology design. Use topology designs for Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) deployments that are enabled via Chef, HP Server Automation, and HP Operations Orchestration flow-based components.

Each topology design component binds to a single provider for fulfillment automation. Topology designs delegate component lifecycle provisioning to providers.

For More Information

The following links provide more information about software products in this document.

| | |
|---------------------------------------|--|
| HP CSA (HP Passport sign-in required) | HP CSA System and Software Support Matrix HP CSA Documentation List |
| Amazon Web Services (AWS) | AWS Documentation |
| Chef Server | Chef documents |
| VMware vCenter Operations Manager | VMware vCenter Operations Manager Documentation |

To access other toolkits to design and extend cloud services running on HP CloudSystem, go to <http://www.hp.com/go/csdevelopers>.

For more information about HP CloudSystem, visit <http://www.hp.com/go/cloudsystem>

HP software product documentation can be found at <https://softwaresupport.hp.com/>

You need to sign-in or register to use this site. Use the **Search** function at the top of the page to find documentation, whitepapers, and other information sources.

To help us improve our documents, please send feedback to clouddocs@hp.com.