



CHAPTER 48

Overview of Templates and Data Files

Templates provide a means to deploy commands and configurations not normally supported by Cisco Prime Fulfillment to a device. Templates are written in the Velocity Template Language (VTL) and are generally comprised of IOS and IOS XR device CLI configurations.

Templates support the browsing, creation, and deletion of Template Folders, Templates, and Data Files and it supports the viewing of Template-generated configurations. This is applicable to both IOS and IOS XR. For IOS XR devices the configlet generated from template datafiles are CLI commands, not XML commands.

The configuration created from the template and data file can be downloaded to devices. When creating a Service Request, you can select from the list of templates and data files and associate them with the Service Request. At Deploy time, the template and data file are instantiated and the configuration is appended or prepended to the configlet generated by Prime Fulfillment. Another method is to use the Device Console feature to download templates independent of Service Requests, as explained in the [“Download Template” section on page 66-3](#).

Prime Fulfillment provides a way to integrate a template with Prime Fulfillment configlets.

For a given customer edge router and/or provider edge router, you specify the following:

- template name
- template data file name
- whether the template configuration file should be appended or prepended to the Prime Fulfillment configlet
- whether the template configuration file is active or inactive for downloading to the edge device

The template data files are tightly linked with the corresponding template (a data file cannot be linked to more than one template). You can use a data file and its associated template to create a template configuration file. The template configuration file is merged with (either appended or prepended to) the Prime Fulfillment configlet. Prime Fulfillment downloads the combined Prime Fulfillment configlet and template configuration file to the edge device router.

- You can download a template configuration file to a router.
- You can apply the same template to multiple edge routers, assigning the appropriate template data file for each device. Each template data file includes the specific data for a particular device (for example, the management IP address or hostname of each device).

Template commands are treated independently from those associated with a service creation (Multi Protocol Label Switching (MPLS), Layer 2 Virtual Private Network (L2VPN), Virtual Private LAN Service (VPLS), Traffic Engineering (TE), and so on). Consequently, template commands must be removed separately from the device(s) during a service decommission. To remove prior template commands, a separate template is needed during a decommission process. Decommissioning a service

request does not automatically remove the original template commands. A separate negate template needs to be added to the decommission process and the original templates must be removed. The negate template must contain the necessary NO commands to successfully remove any unwanted IOS commands added by the original template.

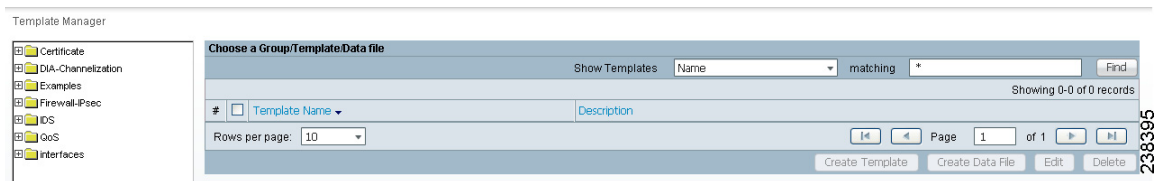
**Note**

For additional information about template usage, see the [Template Usage, page 48-40](#).

To use Templates, follow these steps:

- Step 1** Choose **Service Design > Templates > Template Manager** and you receive a window as shown in [Figure 48-1](#),

Figure 48-1 *Templates Manager*



Template examples are shown in the left column. A complete list of template examples is specified in the [Template Examples, page 48-18](#). A complete list of Repository variables is shown in the “[Summary of Repository Variables](#)” section on [page 48-20](#). An explanation of a tool for importing and exporting templates into and from an Prime Fulfillment database is given in the “[Importing and Exporting Templates](#)” section on [page 48-38](#).

- Step 2** Then you can do any of the following:
- [View Templates Tree and Data Pane, page 48-2](#)
 - [Create Folders and Subfolders, page 48-3](#)
 - [Create Template, page 48-3](#)
 - [Create Data File, page 48-14](#)
 - [Edit, page 48-16](#)
 - [Delete, page 48-17](#)

View Templates Tree and Data Pane

When you choose **Service Design > Templates > Template Manager**, you receive a window as shown in [Figure 48-1](#).

The Templates tree is in the left column. You can continue clicking the + sign next to each created folder and subfolder until you get to the last level of information. The last possible level is the template name. Data file information is not kept in the tree.

The right section of the window is the data pane. The name of the folder or template is in the upper-left corner. When you check the check box next to the template or data file information, the **Create Template**, **Create Data File**, **Edit**, or **Delete** buttons are enabled as described in the following sections.

When there are many templates in a folder or many data files in a template, the **Show Templates matching** or **Show Data Files matching** filter in the upper right-hand corner of the data pane can be very useful. For example, you can click the drop-down list for **Show Templates** or **Show Data Files** and choose to match (matches are case-sensitive) the **Name** or **Description** and then in the **matching** box you can choose to work with templates or data files, respectively, that start with **abc**. In this case, enter **abc*** in the field and then click the **Show** button. Only the templates or data files, respectively, that start with **abc** appear. For more information about filters, see [Filters, page 1-5](#).

**Note**

The template search facility applies to the folder currently selected and not across all folders.

**Note**

The data file search applies to the template currently selected and not across all folders and templates.

You can also **View** configurations when the table displays data files.

Create Folders and Subfolders

To create a new folder or subfolder, follow these steps:

Step 1 Choose **Service Design > Templates > Template Manager**.

Step 2 In the **Template Manager** tree, right-click in the white area and choose **New > Folder** to create a new folder or right-click on an existing folder or subfolder and choose **New > Folder** to create a subfolder.

**Note**

There is no limit to the number of levels of folders and subfolders you can create.

Step 3 In the new text field that appears in the **Template Manager** tree, enter the new folder or subfolder name.

Copying Folders or Subfolders

To copy a folder or subfolder and paste it into another folder or subfolder, follow these steps:

Step 1 Choose a folder or subfolder and then right-click and you receive the opportunity to copy. Click **Copy**.

Step 2 Right-click on the folder or subfolder into which you want to paste the copied folder or subfolder and all its content and click **Paste**.

You will see the new folder or subfolder and all its content in the selected location. You can edit from there.

Create Template

You can either create a new template in an existing folder or you can create a new folder first and then create the template. To create a new folder, see the section [“Create Folders and Subfolders”](#).

To create a new template, follow these steps:

- Step 1** Choose **Service Design > Templates > Template Manager**.
- Step 2** In the **Template Manager** tree, click on the folder in which you want to create a new template. A window appears as shown in [Figure 48-2](#).

Figure 48-2 Folder with Existing Templates

The screenshot shows a window titled "Folder: DIA-Channelization". It has a "Show Templates" section with a "Name" dropdown and a "matching" field containing an asterisk. Below this is a table with 7 records. The table has columns for "#", "Template Name", and "Description". The records are:

#	Template Name	Description
1	PA-MC-T3-CHANNELIZED	
2	PA-MC-STM1-AU4-CHANNELIZED	
3	PA-MC-STM1-AU3-CHANNELIZED	
4	PA-MC-E3-CHANNELIZED	
5	10K-CT3-UNCHANNELIZED	
6	10K-CT3-CHANNELIZED	
7	10K-CHOC12-STS1-PATH	

At the bottom, there are controls for "Rows per page" (set to 10), "Page 1 of 1", and buttons for "Create Template", "Create Data File", "Edit", and "Delete".

- Step 3** You can use the **Show Templates** drop-down list to choose whether to view the templates alphabetically by **Name** or by **Description**. Then click the **Show** button to activate how you view the templates. If you enter characters in the **matching** field before you click the **Show** button, you minimize the list of templates that appear either by **Name** or by **Description**. For more details, see [View Templates Tree and Data Pane, page 48-2](#).
- Step 4** Click the **Create Template** button and you receive a window as shown in [Figure 48-3](#).

Figure 48-3 Template Editor

The screenshot shows a "Template Editor" window. It has a "Template Information" section with the following fields:

- Template Name ***: A text input field.
- Description**: A text input field.
- Body ***: A large text area for the template body.

Below these fields are two checkboxes:

- Has Negate Template**: ☐
- Has User Reference**: ☐

At the bottom, there are buttons for "Select", "Save", and "Close". A note at the bottom left states: "Note: * - Required Field".

- Step 5** Enter the following:
- Template Name** (required)—This must be a unique name within a folder. This name must begin with an alphabetic character and can only contain alphanumeric characters, underscores, and hyphens.
 - Description** (optional) —You can enter any description here.

- **Body** (required)—Enter the configuration text, Velocity Template Language (VTL) directives, and variables that you want included.

**Note**

The VTL is the mark-up language used to describe the template. The VTL is explained at <http://velocity.apache.org>. For more specific information, you might like to navigate to <http://velocity.apache.org/engine/devel/user-guide.html> or <http://velocity.apache.org/engine/devel/vtl-reference-guide.html>.

**Note**

For additional information about template usage, see the [Template Usage, page 48-40](#).

Step 6

Click the **Select** drop-down list, and choose from the following:

- [Negate Template, page 48-5](#)
- [User Reference, page 48-6](#)
- [Optional Attributes, page 48-7](#)
- [Sub-Template, page 48-9](#)
- [Variables, page 48-10](#)
- [Validate, page 48-13](#)

Negate Template

To remove a configuration created from a template or datafile, you must apply **Negate** to the existing service. The negate template is saved as `<TemplateName>.Negate` in the same folder as the original template. When a template is removed, the negate template is also deleted. You can also delete the negate template separately. Datafiles can be associated for the negate template.

When a template is associated in a service Policy and Service Request, the negate template is automatically associated (see the [Cisco Prime Fulfillment User Guide 6.1](#)).

During decommissioning, a negate template is used for deployment. If you change a template, the negate template automatically changes to the negate template of the newly selected template.

Do the following after clicking the **Select** drop-down list in [Step 6](#) of the “[Create Template](#)” section:

Step 1

Choose **Negate** and then click the **Go** button and you receive a window as in [Figure 48-4](#).

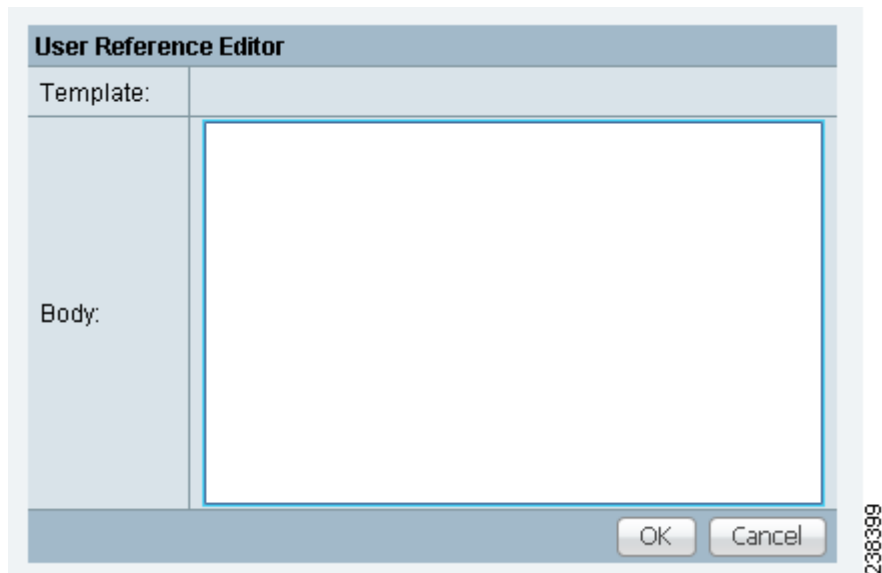
Figure 48-4 *Negate Template Editor*

- Step 2** Optionally add the name of the negate template in **Description**.
- Step 3** Enter the template information in the required Body block. Enter **no** to indicate negate before each line of information, corresponding to the lines in the template.

User Reference

You can keep information about this template by using **User Reference**.
Do the following after clicking the **Select** drop-down list in [Step 6](#) of the “[Create Template](#)” section:

- Step 1** Choose **User Reference** and then click the **Go** button and you receive a window as in [Figure 48-5](#).

Figure 48-5 User Reference Editor


The **User Reference Editor** dialog box has a title bar with the same name. It contains two main sections: **Template:** and **Body:**. The **Body:** section is a large, empty rectangular area with a blue border. At the bottom right of the dialog are **OK** and **Cancel** buttons. A vertical text label "238399" is positioned to the right of the dialog box.

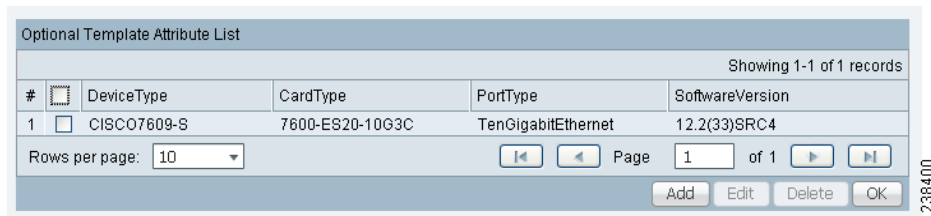
- Step 2** In [Figure 48-5](#), you can add information in the available fields, **Template** and **Body**.
- Step 3** When you click the **OK** button, the information updates in [Figure 48-3](#). When you click **Cancel**, you return to [Figure 48-3](#) without updates.

Optional Attributes

When you choose **Optional Attributes**, you can view the predefined **Device Type**, **Card Type**, **Port Type**, and **Software Version** (IOS and IOS XR) populated from the Prime Fulfillment repository. When no attribute value is provided for any of the four categories, the attribute is applicable for all in that type. For example, if the drop-down list for **Port Type** has no choices, the attribute value is applicable for all Port Types. Each combination of attributes should match. Each combination of attributes is called an attribute set, and templates can have multiple attributes, for example, a template can be applicable for the 7600 series and the 3500 series.

Do the following after clicking the **Select** drop-down list in [Step 6](#) of the “**Create Template**” section:

- Step 1** Choose **Optional Attributes** and then click the **Go** button and you receive a window as in [Figure 48-6](#).

Figure 48-6 Optional Template Attribute List


The **Optional Template Attribute List** window displays a table of attributes. The title bar says "Optional Template Attribute List". Below the title bar, it says "Showing 1-1 of 1 records". The table has five columns: #, ☐, DeviceType, CardType, PortType, and SoftwareVersion. The first row contains the values: 1, ☐, CISCO7609-S, 7600-ES20-10G3C, TenGigabitEthernet, and 12.2(33)SRC4. Below the table, there is a "Rows per page:" dropdown set to "10", and navigation buttons: <<, <, Page, 1, of 1, >, >>. At the bottom right are **Add**, **Edit**, **Delete**, and **OK** buttons. A vertical text label "238400" is positioned to the right of the window.

#	<input type="checkbox"/>	DeviceType	CardType	PortType	SoftwareVersion
1	<input type="checkbox"/>	CISCO7609-S	7600-ES20-10G3C	TenGigabitEthernet	12.2(33)SRC4

- Step 2** You can view the predefined **Device Type**, **Card Type**, **Port Type**, and **Software Version** (IOS and IOS XR) populated from the Prime Fulfillment repository. When no attribute value is provided for any of the four categories, the attribute is applicable for all in that type. Templates can have multiple attributes. You are required to create different templates based on roles and associate them to a Policy and Service Request (see the [Cisco Prime Fulfillment User Guide 6.1](#)).
- Step 3** Check the check box for the attribute set (row of information) for which you want to do the following (except for **Add**, when you should not check a check box):
- Click the **Add** button to open the optional attributes editor for adding attributes. The added attribute set is then reflected in the attribute list page.
 - Click the **Edit** button to open the optional template attributes editor for modifying attributes. Multiple editing in one process is not allowed.
 - Click the **Delete** button and the selected attributes are deleted. You can delete multiple selected attributes at the same time.
 - Click the **OK** button and the window closes and you return to the previous page.
- Step 4** When you click the **Add** or **Edit** button, a popup window appears in which you can enter the optional identifiers, as shown in [Figure 48-7](#).



Note Before clicking the **Edit** button, you must check the check box for the one attribute set (row of information) in [Figure 48-6](#) that you want to edit. You cannot edit multiple rows at the same time.

Figure 48-7 Optional Template Attributes Editor

Optional Template Attributes Editor	
Device Type:	Select One
Software Version:	Select One
Card Type:	Select One
Port Type:	Select One
<div>Reset Refresh OK Cancel</div>	

- Step 5** In [Figure 48-7](#), click the drop-down list for each of **Device Type**, **Software Version**, **Card Type**, and **Port Type**.



Note The drop-down lists are intelligently filtered based on selection in the previous attribute. For example, if you have selected the 7600 for the **Device Type**, then the **Card Type** choices are related to the 7600.

- Step 6** Click one of the following buttons:
- Reset**—Allows you to start over in this selection process.
 - Refresh**—Refreshes the option list from the database and from the user-defined file. The user-defined attributes are read from the **usertemplateattr.xml** file.

**Note**

The user-defined attribute file name **usertemplateattr.xml** can be changed by using the DCPL property: **TemplateManger\userTemplateAttrFile**. (See [Appendix B, “Property Settings”](#) for more details.)

**Note**

The **Refresh** process can take some time. Just be aware of this.

- **OK**—Accepts your selected template attributes, adds them as a set, and returns you to an updated [Figure 48-6](#) with an added attribute set (row of information).
- **Cancel**—Returns you to the previous window without any changes.

Sub-Template

A template using other templates is called a super-template. The template being used is called the sub-template. The super-template instantiates all required sub-templates by passing values for the variables in the sub-template. After instantiation, the super-template puts the sub-template generated configlet into the super-template.

Do the following after clicking the **Select** drop-down list in [Step 6](#) of the “[Create Template](#)” section:

- Step 1** Choose **Sub-Template** and then click the **Go** button and you receive a window as in [Figure 48-8](#).

Figure 48-8 Sub-Template Editor

- Step 2** Check the check box for the sub-template (row of information) for which you want to do the following (except for **Add**, when you should not check a check box):
- Click the **Add** button to add a new row. Then under the **Sub Templates** column, click **Add link** and a new pop-up appears from which you can choose the new subtemplates. Default check boxes are unselected. The changes are not persisted until saved by clicking the **Ok** button.
 - Click the **Delete** button to delete selected rows. You can delete multiple selected rows at the same time. The changes are not persisted until saved by clicking the **Ok** button.
 - Click the **OK** button and all changes will be saved on the form. The window closes and you return to the previous page.
 - Click the **Cancel** button and all the changes are discarded. The window closes and you return to the previous page.

Step 3 You can associate a sub-template with a super-template. When the templates are instantiated during service provisioning (see the [Cisco Prime Fulfillment User Guide 6.1](#)), the appropriate sub-templates are used based on the run time information on the device, line card, role, port, and device software versions. Appropriate sub-template attributes provided by the user are instantiated during deployment based on the attributes. The following are some points to be aware of:

- Only one level of sub-template is supported, but there are no checks for depth of sub-templates.
- No validations occur to check if super-template and sub-template structure is cyclic.
- When you try to delete a sub-template that is referenced by a super-template, a warning message appears. You can modify a sub-template.
- Sub-templates can be attached to multiple super-templates.
- Datafiles are not supported for sub-templates. If multiple datafiles are found, the first available datafile is chosen based on the alphabetic sorting during deployment.

Step 4 You can mark a sub-template as default. There will be a default for the **Device** type and the **Software** version attribute types. When no attributes are marked for the templates, the template is treated as a default template. These templates have lower preference than default sub-templates for an attribute type. When multiple subtemplates have no attributes marked, no subtemplate is selected. When a template is being matched, the order in [Table 48-1](#) is the order followed for matching defaults.

Table 48-1 Matching Defaults

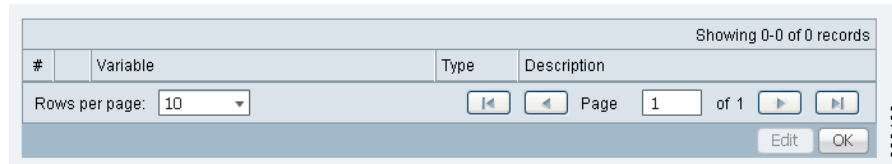
Matching Order	Role	Device	Line Card	Port	Software
1	Exact Match	Exact Match	Exact Match	Exact Match	Exact Match
2	Exact Match	Exact Match	Exact Match	Exact Match	Previous Highest
3	Exact Match	Exact Match	Exact Match	No Values	Exact Match
4	Exact Match	Exact Match	Exact Match	No Values	Previous Highest
5	Exact Match	Exact Match	No Values	No Values	Exact Match
6	Exact Match	Exact Match	No Values	No Values	Previous Highest
7	Exact Match	Exact Match	No Values	No Values	No Values
8	Exact Match	No Values	Exact Match	Exact Match	Exact Match
9	Exact Match	No Values	Exact Match	Exact Match	Previous Highest
10	Exact Match	No Values	Exact Match	No Values	Exact Match
11	Exact Match	No Values	Exact Match	No Values	Previous Highest
12	Exact Match	No Values	No Values	No Values	Exact Match
13	Exact Match	No Values	No Values	No Values	Previous Highest
14	Exact Match	Default	No Values	No Values	No Values
15	Exact Match	No Values	No Values	No Values	Default
16	Exact Match	No Values	No Values	No Values	No Values

Variables

Do the following after clicking the **Select** drop-down list in [Step 6](#) of the “Create Template” section:

Step 1 Choose **Variables** and then click the **Go** button and you receive a window as in [Figure 48-9](#).

Figure 48-9 *Template Variables*



Step 2 Click the radio button for the Variable you want to edit and click **Edit**.

You receive a Variable Definition window.

Step 3 Click the drop-down list for **Type** to receive the following choices:

- **String**—Proceed to [Step 4](#).
- **Integer**—Proceed to [Step 5](#).
- **Float**—Proceed to [Step 6](#).
- **IPv4 Address**—Proceed to [Step 7](#).
- **Sub-Template**—Proceed to [Step 8](#).

Step 4 The default Type to appear is **String**, a combination of ASCII characters considered as a group. The resulting Variable Definition window for Type String is shown and its attributes are as follows:

- **Description** (optional)—You can enter any descriptive statement about this variable here.
- **Required**—Leave the default of the checked check box if this variable is required. Otherwise, uncheck it.
- **Dimension**—Choose **0** (default), which indicates a scalar or enum variable; choose **1**, in which case the variable becomes a one-dimensional array; or choose **2**, in which case the variable becomes a two-dimensional array.
- **Pattern** (optional)—Specify a regular expression pattern of the string. For example, a pattern of **isc[0-9]+** defines a string that starts with **isc** followed by one or more digits from **0** to **9**.
- **Minimum Length** (optional) —f you specify a minimum length, the string cannot be less than the length specified here.
- **Maximum Length** (optional)—If you specify a maximum length, the string cannot exceed the length specified here.
- **Default** radio button (optional)—If there is a default value for the specified variable, specify it here.
- **Available Values** radio button (optional)—Enter string values for this variable. Separate the values by commas.

After you enter all the data, click **OK** to accept this information for the specified variable; continue editing all variables you want to change in this same way, then click **OK** in a window such as [Figure 48-9](#), which now includes these updated variables; click **Save** and then **Close** or click **Close** and when asked, agree to **Save** for a window such as [Figure 48-3](#). Create a Data File is shown in the “[Create Data File](#)” section on page 48-14, **Edit** is shown in the “[Edit](#)” section on page 48-16, and **Delete** is shown in the “[Delete](#)” section on page 48-17.

Step 5 When you choose the Type **Integer**, a whole number, the resulting Variable Definition window for Type Integer is shown and its attributes are as follows:

- **Description** (optional)—You can enter any descriptive statement about this variable here.

- **Required**—Leave the default of the checked check box if this variable is required. Otherwise, uncheck it.
- **Dimension**—Choose **0** (default), which indicates a scalar or enum variable; choose **1**, in which case the variable becomes a one-dimensional array; or choose **2**, in which case the variable becomes a two-dimensional array.
- **Minimum Value** (optional)—If you specify a minimum value, the integer cannot be less than the value specified here.
- **Maximum Value** (optional)—If you specify a maximum value, the integer cannot exceed the value specified here.
- **Default** radio button (optional)—If there is a default value for the specified variable, specify it in the field after the radio button.
- **Available Values** radio button (optional)—Enter string values for this variable in the field after the radio button. Separate the values by commas.

After you enter all the data, click **OK** to accept this information for the specified variable; continue editing all variables you want to change in this same way, then click **OK** in a window such as [Figure 48-9](#), which now includes these updated variables; click **Save** and then **Close** or click **Close** and when asked, agree to **Save** for a window such as [Figure 48-3](#). Create a Data File is shown in the “[Create Data File](#)” section on page 48-14, **Edit** is shown in the “[Edit](#)” section on page 48-16, and **Delete** is shown in the “[Delete](#)” section on page 48-17.

Step 6 When you choose the Type **Float**, a number that has no fixed number of digits before or after the decimal point, the resulting Variable Definition window for Type Float is shown and its attributes are as follows:

- **Description** (optional)—You can enter any descriptive statement about this variable here.
- **Required**—Leave the default of the checked check box if this variable is required. Otherwise, uncheck it.
- **Dimension**—Choose **0** (default), which indicates a scalar or enum variable; choose **1**, in which case the variable becomes a one-dimensional array; or choose **2**, in which case the variable becomes a two-dimensional array.
- **Minimum Value** (optional)—If you specify a minimum value, the floating point value cannot be less than the value specified here.
- **Maximum Value** (optional)—If you specify a maximum value, the floating point value cannot exceed the value specified here.
- **Default** radio button (optional)—If there is a default value for the specified variable, specify it here.
- **Available Values** radio button (optional)—Enter string values for this variable. Separate the values by commas.

After you enter all the data, click **OK** to accept this information for the specified variable; continue editing all variables you want to change in this same way, then click **OK** in a window such as [Figure 48-9](#), which now includes these updated variables; click **Save** and then **Close** or click **Close** and when asked, agree to **Save** for a window such as [Figure 48-3](#). Create a Data File is shown in the “[Create Data File](#)” section on page 48-14, **Edit** is shown in the “[Edit](#)” section on page 48-16, and **Delete** is shown in the “[Delete](#)” section on page 48-17.

Step 7 When you choose the Type **IPv4 Address**, the resulting Variable Definition window for Type IPv4 Address is shown and its attributes are as follows:

- **Description** (optional)—You can enter any descriptive statement about this variable here.
- **Required**—Leave the default of the checked check box if this variable is required. Otherwise, uncheck it.

- **Dimension**—Choose **0** (default), which indicates a scalar or enum variable; choose **1**, in which case the variable becomes a one-dimensional array; or choose **2**, in which case the variable becomes a two-dimensional array.
- **Subnet Mask** (optional)—Enter a valid subnet mask.
- **Class** (optional)—Enter the class of the IP address. The options are: **Undefined**, **A**, **B**, or **C**.
- **Default** radio button (optional)—If there is a default value for the specified variable, specify it here.
- **Available Values** radio button (optional)—Enter string values for this variable. Separate the values by commas.

After you enter all the data, click **OK** to accept this information for the specified variable; continue editing all variables you want to change in this same way, then click **OK** in a window such as [Figure 48-9](#), which now includes these updated variables; click **Save** and then **Close** or click **Close** and when asked, agree to **Save** for a window such as [Figure 48-3](#). Create a Data File is shown in the “[Create Data File](#)” section on page 48-14, **Edit** is shown in the “[Edit](#)” section on page 48-16, and **Delete** is shown in the “[Delete](#)” section on page 48-17.

Step 8 When you choose the Type **Sub-Template**, you instantiate one subtemplate into the Main template. The resulting Variable Definition window for Type Sub-Template is shown and its attributes are as follows:

- **Description** (optional)—You can enter any descriptive statement about this variable here.
- **Required**—Leave the default of the checked check box if this variable is required. Otherwise, uncheck it.
- **Location** (required)—Enter the full path name of the parent template. For example `/test2/testyy`.

The variable `varName` is defined as the subtemplate type (by selecting **Variables** and clicking **Go**). The Sub-Template defined earlier is called and you must provide the subtemplate path. The syntax is as follows:

```
$<varName>.callWithDatafile(<DatafileName>)
```

After you enter all the data, click **OK** to accept this information for the specified variable; continue editing all variables you want to change in this same way, then click **OK**, which now includes these updated variables; click **Save** and then **Close** or click **Close** and when asked, agree to **Save** for a window such as [Figure 48-3](#). Create a Data File is shown in the “[Create Data File](#)” section on page 48-14, **Edit** is shown in the “[Edit](#)” section on page 48-16, and **Delete** is shown in the “[Delete](#)” section on page 48-17.

Validate

To validate the information you entered in [Figure 48-3](#) (see [Step 5](#)), do the following after clicking the **Select & Click Go** drop-down list in [Step 6](#) of the “[Create Template](#)” section:

-
- Step 1** Choose **Validate** and then click the **Go** button.
- Step 2** For a successful validation, you will receive a information window appears.
-

Copying Templates

To copy a user-created template and paste it into another folder, follow these steps:

-
- Step 1** Choose a user-created template, not a default template, and then right-click and you receive the opportunity to copy. Click **Copy**.
 - Step 2** Right-click on the folder into which you want to paste the copied template and all its data files and click **Paste**.
 - Step 3** You will see the new template and all its data files in the selected location. You can edit from there.
-

Deleting Templates

To delete a template from a folder, choose a template and then right-click and you receive the opportunity to delete. Click **Delete**.

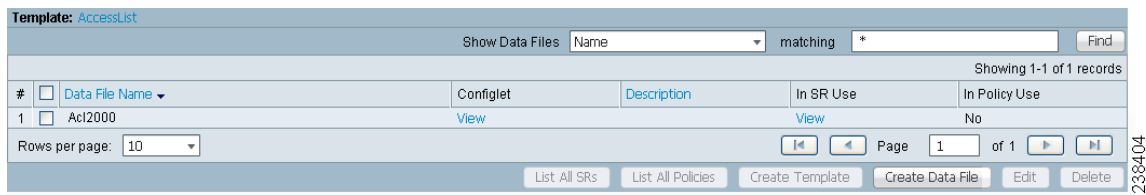
Create Data File

You can create a new data file from an existing template. If the template you want is not available, go to the [“Create Template” section on page 48-3](#).

To create a data file, follow these steps:

-
- Step 1** Choose **Service Design > Templates > Template Manager**.
 - Step 2** In the **Template Manager** tree in the left part of your window, do one of the following
 - 1. Left-click on the folder or subfolder in which the template for which you want to create a data file exists or
 - 2. Click on the **+** next to the folder of choice and then click on the template for which you want to create a data file.
 - Step 3** If you chose 1. in [Step 2](#), a window appears as shown in [Figure 48-2](#).
Check the check box for the template for which you want to create a data file and click **Create Data File**. Then proceed to .
Otherwise, proceed to [Step 4](#).
 - Step 4** If you chose 2. in [Step 2](#), the buttons appear as shown in [Figure 48-10](#).

Figure 48-10 Choose Existing Template, Another Way



Click **Create Data File**. An example of a window that appears is shown in [Figure 48-11](#).

Figure 48-11 Template Data File Editor

Step 5 In the **General** area, fill in the following:

- **Data File Name** (required)—This must be a unique name. This name must begin with an alphabetic character and can only contain alphanumeric characters and the underscore.
- **Description** (optional)—Enter any description that helps you identify this data file.

In the example in [Figure 48-11](#), in the **Variables** area, **ctrlName** is a string variable (**Dimension** defined when the template was created was **0**); you can also create a one-dimensional array (**Dimension** defined when the template was created was **1**); and **t1-list** is a two-dimensional array (**Dimension** defined when the template was created was **2**).

If **t1-list** is a Dynamic Java Class variable, you *must* enter the entire Java Class package name. For example: com.cisco.isc.class_name.



Note

ctrlName can *only* be a string variable.

Step 6 If you click **Vars** as shown in [Figure 48-11](#), you receive a window as shown in [Figure 48-12](#).

Figure 48-12 Template Data File Editor

Click the **Services** drop-down list to have access to variables for:

- **MPLS**
- **L2VPN**
- **VPLS**

- VRF
- FlexUNI

Then click the entry in **Variables** that you want to use and click **Select**.

If you have a **0** dimensional entry (set as **Dimension 0** when creating a template), you can only enter variables in the provided field.

- Step 7** When you click **Edit**, as shown in [Figure 48-11](#), the resulting window depends on whether you are editing a **1** or **2** dimensional array.
- Proceed to [Step 8](#) for information about a **1** dimensional array.
- Proceed to [Step 11](#) for information about a **2** dimensional array.
- Step 8** For a one-dimensional array (set as **Dimension 1** when creating the template), when you click **Edit**, you receive a window.
- Step 9** To add a variable, click **Add** and a window appears in which you can add the variable. Then click **OK**.
- Step 10** To edit or delete a variable, highlight the variable and click **Edit** or **Delete**. For **Edit**, you receive a window appears. Then click **OK**. For **Delete**, *be sure* you want to delete. After you click **Delete**, it automatically occurs and the window is updated. Proceed to [Step 16](#).
- Step 11** For a two-dimensional array (set as **Dimension 2** when creating the template), when you click **Edit**, you receive a window appears.
- Step 12** Click **Add Row** and a window appears. Enter a value and click **OK**.
- Step 13** Click **Add Column** and a window appears.
- Step 14** Enter a value and click **OK**. A resulting window appears.
- Step 15** You can check any of the check boxes (toggles) and you can then **Edit** or **Delete** that row or column. You can also continue to **Add Row** and **Add Column** as shown in [Step 13](#) and [Step 14](#), respectively.
- Step 16** When you complete setting up your two-dimensional array, click **OK**. A window as shown in [Figure 48-11](#) is updated to reflect the new data file information.
- Step 17** You can then click **Save** and then **Close** to save this information and close this file; click **Configure** to show the configuration file; or click **Close** and then be sure to click **OK**, if you want to save the information you have created. If you do not want to save this information, click **Close** and then click **Cancel**.
-

Edit

To edit a Template or Data File, follow these steps:

- Step 1** Choose **Service Design > Templates > Template Manager**.
- Step 2** In the **Template Manager** tree, left-click on the folder or subfolder in which the template you want to edit exists or the template in which the data file you want to edit exists. Alternatively, when the name in the upper left corner of the data pane is a template, you can click on the template name to edit the template.

To edit a template, a window appears as shown in [Figure 48-2](#). To edit a data file, a window appears as shown in [Figure 48-10](#).

Step 3 You can use the **Show Templates** or **Show Data Files** drop-down list to choose whether to view the templates or data files alphabetically by **Name** or by **Description**. Then click the **Show** button to activate how you view the templates or data files. If you enter characters in the **matching** field before you click the **Show** button, you minimize the list of templates or data files that appear either by **Name** or by **Description**. For more details, see the **Show Templates matching** or **Show Data Files matching** filter in the upper right-hand corner of the data pane can be very useful. For example, you can click the drop-down list for **Show Templates** or **Show Data Files** and choose to match (matches are case-sensitive) the **Name** or **Description** and then in the **matching** box you can choose to work with templates or data files, respectively, that start with **abc**. In this case, enter **abc*** in the field and then click the **Show** button. Only the templates or data files, respectively, that start with **abc** appear. For more information about filters, see [View Templates Tree and Data Pane, page 48-2](#).

Step 4 Check the check box for the template or data file you want to edit.

**Note**

For a data file, there is a **Configlet** column in which you can click **View** to view the configuration file.

Step 5 Click **Edit**.

Step 6 When editing a template, you receive a window as shown in [Figure 48-3](#). Then proceed as in [Step 5](#) in the [Create Template](#) section. When editing a data file, you receive a window as shown in [Figure 48-10](#). Then proceed as in [the Create Data File section](#).

Delete

To delete a Template or Data File, follow these steps:

Step 1 Choose **Service Design > Templates**.

Step 2 In the **Templates** tree, left-click on the folder or subfolder in which the template you want to delete exists or the template in which the data file you want to delete exists.

To delete a template, a window appears as shown in [Figure 48-2](#). To delete a data file, a window appears as shown in [Figure 48-10](#).

Step 3 You can use the **Show Templates** or **Show Data Files** drop-down list to choose whether to view the templates or data files alphabetically by **Name** or by **Description**. Then click the **Show** button to activate how you view the templates or data files. If you enter characters in the **matching** field before you click the **Show** button, you minimize the list of templates or data files that appear either by **Name** or by **Description**. For more details, see the **Show Templates matching** or **Show Data Files matching** filter in the upper right-hand corner of the data pane can be very useful. For example, you can click the drop-down list for **Show Templates** or **Show Data Files** and choose to match (matches are case-sensitive) the **Name** or **Description** and then in the **matching** box you can choose to work with templates or data files, respectively, that start with **abc**. In this case, enter **abc*** in the field and then click the **Show** button. Only the templates or data files, respectively, that start with **abc** appear. For more information about filters, see [View Templates Tree and Data Pane, page 48-2](#).

Step 4 Check the check box for the template or data file you want to delete.

**Note**

For a data file, there is a **Configlet** column in which you can click **View** to view the configuration file.

Step 5 Click the **Delete** button.

A confirmation window appears prompting you to confirm the deletion. Before deleting a datafile, make sure it is not associated with a service request, by checking that the **In SR Use** column is set to **No**. When deleting a folder or a template, make sure that none of the datafiles they contain are associated with a service request. By clicking **OK**, you continue the deletion, and by clicking **Cancel**, you cancel the deletion.

You receive an updated window as shown in [Figure 48-2](#), or [Figure 48-10](#), with the deleted template or data file no longer available.

List All SRs

In the **In SR Use** column, as shown in [Figure 48-10](#), **Yes** indicates that the data file is in use and **No** indicates that the data file is not in use. If **Yes** appears, you can click on it and you receive a list of all the associated service requests. If **Yes** appears, a **List All SRs** button is enabled in the bottom row. If you click the **List All SRs** button, all the service requests associated with the selected data file(s) appears, as shown in [Figure 48-13](#). If **No** appears in the **In SR Use** column, the **List All SRs** button is disabled.

From [Figure 48-13](#), if you click the **Close** button, the previous window appears.

Figure 48-13 List All SRs

Service Requests using Data File

Showing 0 of 0 records

#	Data File Name	Job ID	State	Type	Op Type	Creator	Customer Name	Policy Name	Last Modified	Description
---	----------------	--------	-------	------	---------	---------	---------------	-------------	---------------	-------------

Rows per page: 10

◀

▶

Page 1 of 1

Auto Refresh: ☒

Close

238407



Note

The only data files listed in the **Data File Name** column are those selected previously by the user to get to this window. The service request might be associated with other data files that are not displayed.

List All Policies

In the **In Policy Use** column, as shown in [Figure 48-13](#), **Yes** indicates that the data file is in use and **No** indicates that the data file is not in use. If **Yes** appears, you can click on it and you receive a list of all the associated policies. If **Yes** appears, a **List All Policies** button is enabled in the bottom row. If you click the **List All Policies** button, all the policies associated with the selected data file(s) appears. If **No** appears in the **In Policy Use** column, the **List All Policies** button is disabled.

If you click the **Close** button for the newly created window, the previous window appears.



Note

The only data files listed in the **Data File Name** column are those selected previously by the user to get to this window. The policy might be associated with other data files that are not displayed.

Template Examples

In the left column, the hierarchy pane, of **Service Design > Templates**, as shown in [Figure 48-1](#), template examples appear. See [Table 48-2](#).

Table 48-2 **Template Examples and Their Descriptions**

Folder	Template	Description
DIA-Channelization	10K-CHOC12-STS1-PATH	Sample template to break down channelized OC12 to STS-1 paths.
	10K-CT3-CHANNELIZED	Sample template creates T1 out of channelized T3 line card.
	10K-CT3-UNCHANNELIZED	Sample template Creates either a fullrate T3 or a subrate T3 interface out of a channelized T3.
	PA-MC-E3-CHANNELIZED	Sample template Creates E1 (channel groups) out of E3.
	PA-MC-STM1-AU3-CHANNELIZE	Sample template Creates E1 (channel groups) out of TUG-2. This template uses AU-3 AUG mapping that further creates TUG-2s.
	PA-MC-STM1-AU4-CHANNELIZE	Sample template Creates E1 (channel groups) out of TUG-2. This template uses AU-4 AUG mapping that creates TUG-3s and TUG-2s.
	PA-MC-T3-CHANNELIZED	Sample template Creates T1 (channel groups) out of T3.
Examples	AccessList	Demonstrates templates with nested repeat loop and multi-dimension variable.
	AccessList1	Demonstrates the simplest template variable substitution.
	CEWanCOS	Demonstrates if-else statements, repeat statements, mathematical expressions, and one-dimensional variables.
QoS/L2/ATM	CLP_Egress	Sample template to demonstrate the setting of qos_group and ATM Cell Loss Priority at the output of an interface.
	CLP_Ingress	Sample template sets MPLS experimental bit of the ATM Cell marked with Cell Loss Priority, at the input of an interface.
QoS/L2/Ethernet	3400_Egress	
QoS/L2/FrameRelay	classification	Sample template to demonstrate the bandwidth reservation based on FrameRelay DLCI value.

Summary of Repository Variables

This section contains the following tables:

- [Table 48-3 on page 48-20](#), “L2VPN Repository Variables”
- [Table 48-4 on page 48-23](#), “MPLS Repository Variables”
- [Table 48-5 on page 48-33](#), “VPLS Repository Variables”
- [Table 48-6 on page 48-35](#), “VRF Repository Variables”
- [Table 48-7 on page 48-36](#), “FlexUNI/EVC Repository Variables”

[Table 48-3](#) provides a summary of the L2VPN Repository variables available from Prime Fulfillment Templates.

Table 48-3 **L2VPN Repository Variables**

Repository Variable	Dimension	Description
AC_Loopback_Address	0	PE loopback address also known as the router ID.
CARD_TYPE	0	Refers to NPE or UNI interface depending on whether the service is implemented with ethernet access.
CE_DLCI	0	DLCI value on CE for Frame Relay encapsulation.
CE_Encap	0	Encapsulation of the CE interface.
CE_Intf_Desc	0	Interface description for the CE interface.
CE_Intf_Main_Name	0	Major interface name for the CE interface.
CE_Intf_Shutdown	0	Shutdown flag for the CE interface.
CE_VCD	0	VCD value on CE for ATM encapsulation.
CE_VCI	0	VCI value on CE for ATM encapsulation.
CE_Vlan_ID	0	VLAN ID on CE for Ethernet encapsulation.
CE_VPI	0	VPI value on CE for ATM encapsulation.
L2VPNCLECeFacingEncapsulation	0	Encapsulation of the UNI.
L2VPNCLECeFacingInterfaceName	0	Name of the UNI.
L2VPNCLEPeFacingEncapsulation	0	Encapsulation of the NNI (should always be dot1q).
L2VPNCLEPeFacingInterfaceName	1	Name of the NNI (uplinks) (the number can be more than 1 in case of a ring topology, hence any array).
L2VPNDFBIT_SET	0	Indicates not to fragment the bit set (for L2TPv3 only).
L2VPNDynamicModeUseDefaults	0	Dynamic session setup using Prime Fulfillment default values (for L2TPv3 only).
L2VPN_intf_main_name	1	The main interface name for a CE or PE port.

Table 48-3 L2VPN Repository Variables (continued)

Repository Variable	Dimension	Description
L2VPNIP_PMTU	0	Enable the discovery of the path MTU for tunneled traffic (for L2TPv3 only).
L2VPNIP_TOS	0	Configure the value of the TOS byte in IP headers of tunneled packets or reflects the TOS byte value from the inner IP header (for L2TPv3 only).
L2VPNIP_TTL	0	Configure the value of the time to live byte in the IP headers (for L2TPv3 only).
L2VPNL2TP_CLASS_NAME	0	The L2TP class name to overwrite the default L2TP class name (for L2TPv3 only).
L2VPNL2TPv3Sequence	0	Specifies the direction in which sequencing of data packets in a pseudo wire is enabled (for L2TPv3 only).
L2VPNLocalCookieHighValue	0	Specifies the last 4 bytes of the value that the peer PE must include in the cookie field of incoming L2TP packets (for L2TPv3 only).
L2VPNLocalCookieLowValue	0	Specifies the first 4 bytes of the value that the peer PE must include in the cookie field of incoming L2TP packets (for L2TPv3 only).
L2VPNLocalCookieSize	0	Specifies the size (0, 4, or 8) of the cookie field of incoming L2TP packets (for L2TPv3 only).
L2VPNLocalHostName	0	Hostname of the N-PE that peers with a remote N-PE in the L2VPN end-to-end wire.
L2VPNLocalLoopback	0	Loopback address of the N-PE that peers with a remote N-PE in the L2VPN end-to-end wire.
L2VPNLocalSessionId	0	Specifies the ID for the local L2TPv3 session (for L2TPv3 only).
L2VPNLocalSwitchLoopBack1	1	The loopback1 for the local switch (for L2TPv3 only).
L2VPNLocalSwitchLoopBack2	1	The loopback2 for the local switch (for L2TPv3 only).
L2VPNRemoteCookieHighValue	1	Specifies the last 4 bytes of the value that this PE must include in the cookie field of incoming L2RP packets (for L2TPv3 only).
L2VPNRemoteCookieLowValue	1	Specifies the first 4 bytes of the value that this PE must include in the cookie field of incoming L2RP packets (for L2TPv3 only).
L2VPNRemoteCookieSize	1	Specifies the size (0, 4, or 8) of the cookie field of outgoing L2TP packets (for L2TPv3 only).

Table 48-3 L2VPN Repository Variables (continued)

Repository Variable	Dimension	Description
L2VPNRemoteHostName	0	Hostname of the remote N-PE that peers with the N-PE in context in the L2VPN end-to-end wire.
L2VPNRemoteLoopback	0	Loopback address of the remote N-PE that peers with the N-PE in context in the L2VPN end-to-end wire.
L2VPNRemoteSessionID	1	Specifies the ID for the remote L2TPv3 session (for L2TPv3 only).
L2VPNSessionSetupMode	0	Defines how the L2TPv3 session is set up (static or dynamic) (for L2TPv3 only).
L2VPNTransportMode	0	Defines how the L2TPv3 data is transferred (for Frame Relay: DLCI or Port; for ATM: VP or VC) (for L2TPv3 only).
L2VPNUniMajorInterfaceName	0	The main interface name of the UNI.
L2VPNVCId	0	The virtual circuit ID of the L2TPv3 or ATOM tunnel.
PE_DLCI	0	DLCI value on PE for Frame Relay encapsulation.
PE_Encap	0	Encapsulation of the PE interface.
PE_Intf_Desc	0	Interface description for the PE interface.
PE_Intf_Main_Name	0	Major interface name for the PE interface.
PE_VCD	0	VCD value on PE for ATM encapsulation.
PE_VCI	0	VCI value on PE for ATM encapsulation.
PE_Vlan_ID	0	VLAN ID on PE for Ethernet encapsulation.
PE_VPI	0	VPI value on PE for ATM encapsulation.
PseudoWire_Class_Type_Of_Core	0	Core type of the Service Provider over which L2VPN is provisioned.
Uni_Aging	0	Length of time the MAC address can stay on the port security table.
Uni_Cdp_Enable	0	Flag to enable or disable layer 2 tunnelling on a Cisco Discover Protocol (CDP).
Uni_Cdp_Threshold	0	Number of packets per second to be received before the interface is shut down for the CDP protocol.
Uni_Mac_Address	0	Number of MAC addresses allowed for port security.
Uni_Port_Security	0	Flag to enable or disable security on a UNI interface.
Uni_Protocol_Tunnelling	0	Flag to enable or disable Layer 2 Bridge Protocol Data Unit (BPDU) protocol tunnelling on a UNI interface.

Table 48-3 L2VPN Repository Variables (continued)

Repository Variable	Dimension	Description
Uni_Recovery_Interval	0	Amount of time to wait before recovering a UNI port.
Uni_Shutdown	0	Flag indicating whether the User Network Interface (UNI) is shutdown.
Uni_Speed	0	Value of the UNI link speed.
Uni_Stp_Enable	0	Flag to enable or disable layer 2 tunnelling on a Spanning Tree Protocol (STP).
Uni_Stp_Threshold	0	Flag to enable or disable layer 2 tunnelling on an STP.
Uni_Violation_Access	0	Action taken when a port security violation is detected.
Uni_Vtp_Enable	0	Flag to enable or disable layer 2 tunnelling on a VLAN Trunk Protocol (VTP).
Uni_Vtp_Threshold	0	Flag to enable or disable layer 2 tunnelling on a VTP.

Table 48-4 provides a summary of the MPLS Repository variables available from Prime Fulfillment Templates.

Table 48-4 MPLS Repository Variables

Repository Variable	Dimension	Description
Advertised_Routes_To_CE	2	List of one or more IP addresses of the advertised static route to be placed on the PE to define the CE's address space.
CARD_TYPE	0	Refers to NPE or UNI interface depending on whether the service is implemented with ethernet access.
CE_BGP_AS_ID	0	BGP AS ID on a CE when the routing protocol between a CE and a PE is BGP.
CE_BGP_AS_ID_IPV6	0	If the Address family is IPv6, this specifies the Border Gateway Protocol (BGP) routing protocol Autonomous System (AS) number.
CE_DLCI	0	DLCI value on CE for Frame Relay encapsulation.
CE_EIGRP_AS_ID	0	EIGRP AS ID on a CE when the routing protocol between a CE and a PE is EIGRP.
CE_Facing_MVRFCE_BGP_AS_ID	0	BGP AS ID on an MVRFCE when the routing protocol between a CE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
CE_Facing_MVRFCE_DLCI	0	DLCI value on CE facing MVRFCE interface for Frame Relay encapsulation, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_EIGRP_AS_ID	0	EIGRP AS ID on an MVRFCE when the routing protocol between a CE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Intf	0	Name of the CE facing interface on an MVRFCE, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Intf_Address	0	IP address assigned to the CE facing MVRFCE interface, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Intf_Encap	0	Encapsulation for CE facing of an MVRFCE interface, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Intf_Name	0	Name of the CE facing MVRFCE interface, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Intf_Type	0	Interface type for CE facing of an MVRFCE interface, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Ospf_Process_ID	0	OSPF process ID on MVRFCE when the routing protocol between a CE and an MVRFCE is OSPF, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_Tunnel_Src_Addr	0	Tunnel source address on CE facing MVRFCE interface for GRE encapsulation when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_VCD	0	VCD value on CE facing MVRFCE interface for ATM encapsulation, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_VCI	0	VCI value on CE facing MVRFCE interface for ATM encapsulation, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_VLAN_ID	0	VLAN ID on CE facing MVRFCE interface for Ethernet encapsulation, when an MPLS link includes an MVRFCE.
CE_Facing_MVRFCE_VPI	0	VPI value on CE facing MVRFCE interface for ATM encapsulation, when an MPLS link includes an MVRFCE.
CE_Intf_Address	0	IP address assigned to the CE interface.
CE_Intf_Encap	0	Encapsulation of the CE interface.
CE_Intf_Name	0	Name of the CE interface.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
CE_MVRFCE_Bandwidth_Metric_For_Redistribution	0	Bandwidth metric for redistribution of EIGRP when the routing protocol between a CE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFC.
CE_MVRFCE_BGP_AS_ID	0	BGP AS ID on a CE when the routing protocol between a CE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.
CE_MVRFCE_Delay_Metric_For_Redistribution	0	Delay metric for redistribution of EIGRP when the routing protocol between a CE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFC.
CE_MVRFCE_EIGRP_AS_ID	0	EIGRP AS ID on a CE when the routing protocol between a CE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
CE_MVRFCE_Loading_Metric_For_Redistribution	0	Loading metric for redistribution of EIGRP when the routing protocol between a CE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFC.
CE_MVRFCE_MTU_Metric_For_Redistribution	0	MTU metric for redistribution of EIGRP when the routing protocol between a CE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFC.
CE_MVRFCE_Ospf_Process_ID	0	OSPF process ID on CE when the routing protocol between a CE and an MVRFCE is OSPF, when an MPLS link includes an MVRFCE.
CE_Ospf_Process_ID	0	OSPF process ID on CE when the routing protocol between a CE and a PE is OSPF.
CE_Tunnel_Src_Addr	0	Tunnel source address on CE for GRE encapsulation.
CE_VCD	0	VCD value on CE for ATM encapsulation.
CE_VCI	0	VCI value on CE for ATM encapsulation.
CE_Vlan_ID	0	VLAN ID on CE for Ethernet encapsulation.
CE_VPI	0	VPI value on CE for ATM encapsulation.
Export_Map	0	Name of the export map associated with the VRF.
Extra_CE_Loopback_Required	0	Flag to indicate whether an extra loopback request is required on the CE.
Import_Map	0	Name of the import map associated with the VRF.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
Is_Default_Info_Originate	0	Flag to indicate whether the default-information originate command for BGP on the PE when STATIC is a running protocol between a CE and a PE.
Is_Default_Info_Originate_IPv6	0	If the Address family is IPv6, Flag to indicate whether the default-information originate command for BGP on the PE when STATIC is a running protocol between a CE and a PE.
Is_Default_Routes_Sent_To_CE	0	Flag to indicate whether the default routes are sent to a remote CE.
Join_Grey_Mgmt_Vpn	0	Flag to indicate whether MPLS will join a Grey Management VPN.
Max_route_threshold	0	Percentage of the maximum number of routes that can be imported into the VRF.
Max_Routes	0	Maximum number of routes than can be imported into the VRF.
MPLSCeInterfaceMask	0	The mask of the IP address assigned to the CE interface for a particular MPLS VPN link.
MPLSCeLoopbackAddress	0	The IP address of the extra CE loopback address for a particular MPLS VPN link.
MPLSCLECeFacingEncapsulation	0	The encapsulation of the interface on the device facing the CE for that particular MPLS VPN link.
MPLSCLECeFacingInterfaceName	0	The name of the interface on the device facing the CE for that particular MPLS VPN link.
MPLSCLEPeFacingEncapsulation	0	The encapsulation of the interface on the device facing the PE for that particular MPLS VPN link.
MPLSCLEPeFacingInterfaceName	0	The name of the interface on the device facing the PE for that particular MPLS VPN link.
MPLSExportRouteTargets	1	List of Route Targets that are exported for a particular VRF associated with the MPLS VPN link.
MPLSImportRouteTargets	1	List of Route Targets that are imported for a particular VRF associated with the MPLS VPN link.
MPLSPeInterfaceMask	0	The mask of the IP address assigned to the PE interface for a particular MPLS VPN link.
Multicast_Enabled_IPv6	0	Enabling and disabling a Multicast IPv6 VPN. If the check box is enabled, Multicast IPv6 VPN configlets are generated.
Multicast_Route_Limit	0	Multicast route limit value for the VRF

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
MVRFCE_CE_Advertised_Routes_To_CE	2	List of one or more IP addresses of the advertised static route to be placed on the PE to define the CE's address space, when the MPLS link includes an MVRFCE.
MVRFCE_CE_IP_Unnumbered	0	Flag to indicate whether the MVRCE to CE link is unnumbered, when an MPLS link includes an MVRFCE.
MVRFCE_CE_Is_Default_routes_Sent_To_CE	0	Flag to indicate whether the default routes are sent to a remote CE, when an MPLS link includes an MVRFCE.
MVRFCE_CE_NBR_ALLOW_AS_IN	0	AllowASIn flag when the routing protocol between a CE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.
MVRFCE_CE_NBR_AS_OVERRIDE	0	ASOverride flag when the routing protocol between a CE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.
MVRFCE_CE_Ospf_Area_Number	0	OSPF area number when the routing protocol between a CE and an MVRCE is OSPF, when an MPLS link includes an MVRFCE.
MVRFCE_CE_Ospf_Route_Policy	0	Name of the Redistribute OSPF route policy to be configured when an MPLS link includes an MVRFCE_CE.
MVRFCE_CE_Routes_To_Reach_Other_Sites	2	List of one or more IP addresses to specify the static routes to put on the CE, when the MPLS link includes an MVRFCE.
MVRFCE_CE_Routing_Protocol	0	Routing protocol between MVRFCE and CE.
PE_BGP_AS_ID	0	BGP AS ID on a PE when the routing protocol between a CE and a PE is BGP.
PE_Cable_Both_Helper_Address_List	1	List of DHCP server IP addresses to which both cable modem and host UDP broadcasts are forwarded.
PE_Cable_Modem_Helper_Address_list	1	List of DHCP server IP addresses to which cable modem UDP broadcasts are forwarded.
PE_Cable_Modem_Host_Helper_Address_List	1	List of DHCP server IP addresses to which host UDP broadcasts are forwarded.
PE_Cable_Modem_Secondary_Address_List	1	List of cable modem secondary addresses for cable interfaces.
PE_CE_Bandwidth_Metric_For_Redistribution	0	Bandwidth metric for redistribution of EIGRP when the routing protocol between a CE and a PE is EIGRP.
PE_CE_BGP_ADVERTISE_INTERVAL_IPV6		Advertising interval value for BGP routing protocol if the Address family is IPv6.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
PE_CE_BGP_DEFAULT_ORIGINATE_ROUTE_POLICY_IPV4	0	Default originate route policy name when the routing protocol between a CE and a PE is BGP.
PE_CE_BGP_DEFAULT_ORIGINATE_ROUTE_POLICY_IPV6	0	Default originate route policy name when the routing protocol between a CE and a PE is BGP, if the address family is IPV6.
PE_CE_BGP_MAX_PREFIX_NUMBER	0	BGPNeighbor MaxPrefix value for BGP routing protocol.
PE_CE_BGP_MAX_PREFIX_NUMBER_IPV6	0	BGPNeighbor MaxPrefix value for BGP routing protocol, if the Address family is IPV6.
PE_CE_BGP_MAX_PREFIX_RESTART	0	BGPNeighborMaxprefix restart value for BGP routing protocol.
PE_CE_BGP_MAX_PREFIX_RESTART_IPV6	0	BGPNeighborMaxprefix restart value for BGP routing protocol, if the address family is IPV6.
PE_CE_BGP_MAX_PREFIX_THRESHOLD	0	BGPNeighborMaxprefix threshold value for BGP routing protocol.
PE_CE_BGP_MAX_PREFIX_THRESHOLD_IPV6	0	BGPNeighborMaxprefix threshold value for BGP routing protocol, if the address family is IPV6.
PE_CE_BGP_MAX_PREFIX_WARNING_ONLY	0	BGPNeighborMaxprefix warnily_only (enable/disable).
PE_CE_BGP_MAX_PREFIX_WARNING_ONLY_IPV6	0	BGPNeighborMaxprefix warnily_only (enable/disable), if the Address family is IPV6.
PE_CE_BGP_Neighbor_Route_Map_Or_Policy_In	0	Name of the BGP Neighbor Route Map/Policy In to be configured on the device.
PE_CE_BGP_Neighbor_Route_Map_Or_Policy_Out	0	Name of the BGP Neighbor Route Map/Policy Out to be configured on the device.
PE_CE_Delay_Metric_For_Redistribution	0	Delay metric for redistribution of EIGRP when the routing protocol between a CE and a PE is EIGRP.
PE_CE_EIGRP_AUTHENTICATION_KEY_CHAIN_NAME	0	Keychain name to authenticate EIGRP protocol traffic on one or more interfaces, if the Routing protocol between CE and PE is EIGRP.
PE_CE_EIGRP_AUTHENTICATION_KEY_CHAIN_NAME_IPV6	0	If the address family is IPV6, this specifies keychain name to authenticate EIGRP protocol traffic on one or more interfaces if the routing protocol between CE and PE is EIGRP

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
PE_CE_IP_Unnumbered	0	Flag to indicate whether the PE to CE link is unnumbered.
PE_CE_IPV6_Routing_Protocol	0	Routing protocol between PE and CE if the address family is IPv6.
PE_CE>Loading_Metric_For_Redistribution	0	Loading metric for redistribution of EIGRP when the routing protocol between a CE and a PE is EIGRP.
PE_CE_MTU_Metric_For_Redistribution	0	MTU metric for redistribution of EIGRP when the routing protocol between a CE and a PE is EIGRP.
PE_CE_NBR_Allow_AS_In	0	AllowASIn flag when the routing protocol between a CE and a PE is BGP.
PE_CE_NBR_Allow_AS_In_IPV6	0	If the Address family is IPv6, AllowASIn flag when the routing protocol between a CE and a PE is BGP.
PE_CE_NBR_AS_Override	0	ASOverride flag when the routing protocol between a CE and a PE is BGP.
PE_CE_NBR_AS_Override_IPV6	0	If the Address family is IPv6, ASOverride flag when the routing protocol between a CE and a PE is BGP.
PE_CE_NBR_Send_Community_IPV6	0	If the Address family is IPv6, then these values specify the “Standard”, “extended”, “Both” of the Send_Community attribute.
PE_CE_Ospf_Area_Number	0	OSPF area number when the routing protocol between a CE and a PE is OSPF.
PE_CE_Ospf_Match_Internal_External	0	Name of the Redistribute OSPF match criteria to be configured on the device.
PE_CE_OSPF_METRIC_TYPE	0	Metric type when the routing protocol between a CE and a PE is OSPF.
PE_CE_OSPF_METRIC_VALUE	0	Metric value when the routing protocol between a CE and a PE is OSPF.
PE_CE_Ospf_Route_Policy	0	Name of the Redistribute OSPF route policy to be configured on the device.
PE_CE_OSPF_ROUTE_POLICY	0	Route policy name when the routing protocol between a CE and a PE is OSPF.
PE_CE_Reliability_Metric_For_Redistribution	0	Reliability metric for redistribution of EIGRP when the routing protocol between a CE and a PE is EIGRP.
PE_CE_Routing_Protocol	0	Routing protocol between PE and CE.
PE_DLCI	0	DLCI value on PE for Frame Relay encapsulation
PE_EIGRP_AS_ID	0	EIGRP AS ID on a PE when the routing protocol between a CE and a PE is EIGRP.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
PE_Facing_MVRFCE_BGP_AS_ID	0	BGP AS ID on an MVRFCE when the routing protocol between a PE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_DLCI	0	DLCI value on PE facing MVRFCE interface for Frame Relay encapsulation, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_EIGRP_AS_ID	0	EIGRP AS ID on an MVRFCE when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_Intf	0	Name of the PE facing interface on an MVRFCE, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_Intf_Address	0	IP address assigned to the PE facing MVRFCE interface, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_Intf_Encap	0	Encapsulation for PE facing of an MVRFCE interface, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_Intf_Name	0	Name of the PE facing MVRFCE interface, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_Intf_Type	0	Interface type for PE facing of an MVRFCE interface, when an MPLS link includes an MVRFCE.
PE_FACING_MVRFCE_OSPF_Process_ID	0	OSPF process ID on an MVRFCE when the routing protocol between a PE and an MVRFCE is OSPF, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_Tunnel_Src_Addr	0	Tunnel source address on PE facing MVRFCE interface for GRE encapsulation when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_VCD	0	VCD value on PE facing MVRFCE interface for ATM encapsulation, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_VCI	0	VCI value on PE facing MVRFCE interface for ATM encapsulation, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_VLAN_ID	0	VLAN ID on PE facing MVRFCE interface for Ethernet encapsulation, when an MPLS link includes an MVRFCE.
PE_Facing_MVRFCE_VPI	0	VPI value on PE facing MVRFCE interface for ATM encapsulation, when an MPLS link includes an MVRFCE.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
PE_Intf_Address	0	IP address assigned to the PE interface.
PE_Intf_Address_IPV6	0	If the Address family is IPv6, this specifies the IP address of the interface.
PE_Intf_Desc	0	Interface description for the PE interface.
PE_Intf_Encap	0	Encapsulation of the PE interface.
PE_Intf_Name	0	Name of the PE interface.
PE_Intf_Shutdown	0	Shutdown flag for the PE interface.
PE_IS_Cable_Modem_Maintenance_Interface	0	Flag to indicate whether the interface is a maintenance interface.
PE_MVRFCE_Bandwidth_Metric_For_Redistribution	0	Bandwidth metric for redistribution of EIGRP when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_BGP_AS_ID	0	BGP AS ID on a PE when the routing protocol between a PE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_Delay_Metric_For_Redistribution	0	Delay metric for redistribution of EIGRP when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_EIGRP_AS_ID	0	EIGRP AS ID on a PE when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_IP_Unnumbered	1	Flag to indicate whether the PE to MVRFCE link is unnumbered, when an MPLS link includes an MVRFCE.
PE_MVRFCE>Loading_Metric_For_Redistribution	0	Loading metric for redistribution of EIGRP when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_MTU_Metric_for_redistribution	0	MTU metric for redistribution of EIGRP when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_NBR_ALLOW_AS_IN	0	AllowASIn flag when the routing protocol between a PE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_NBR_AS_OVERRIDE	0	ASOverride flag when the routing protocol between a PE and an MVRFCE is BGP, when an MPLS link includes an MVRFCE.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
PE_MVRFCE_Ospf_Area_Number	0	OSPF area number when the routing protocol between a PE and an MVRCE is OSPF, when an MPLS link includes an MVRFCE.
PE_MVRFCE_OSPF_Process_ID	0	OSPF process ID on PE when the routing protocol between a PE and an MVRCE is OSPF, when an MPLS link includes an MVRFCE.
PE_MVRFCE_Ospf_Route_Policy	0	Name of the Redistribute OSPF route policy to be configured when an MPLS link includes a PE_MVRFCE.
PE_MVRFCE_Reliability_Metric_For_Redistribution	0	Reliability metric for redistribution of EIGRP when the routing protocol between a PE and an MVRFCE is EIGRP, when an MPLS link includes an MVRFCE.
PE_MVRFCE_Routing_Protocol	0	Routing protocol between PE and MVRFCE, when an MPLS link includes an MVRFCE.
PE_OSPF_PROCESS_ID	0	OSPF process ID on PE when the routing protocol between a CE and a PE is OSPF.
PE_Tunnel_Src_Addr	0	Tunnel source address on PE for GRE encapsulation.
PE_VCD	0	VCD value on PE for ATM encapsulation.
PE_VCI	0	VCI value on PE for ATM encapsulation.
PE_Vlan_ID	0	VLAN ID on PE for Ethernet encapsulation.
PE_VPI	0	VPI value on PE for ATM encapsulation.
rd	0	Route Distinguisher value for the VRF.
RD_FORMAT	0	Defines the RD Format to be used in the MPLS Link, such as RD_AS or RD_IPADDR.
RD_IPADDRESS	0	Defines the RD_IPADDRESS Value to be used in the MPLS Link, if the RD Format is RD_IPADDRESS.
Redistribute_Connected	0	Flag to indicate whether the connected routes are redistributed into BGP on the PE.
Redistribute_Connected_IPV6	0	Flag to indicate whether the connected routes are redistributed into BGP on the PE, if the address family is IPv6.
Redistribute_Static	0	Flag to indicate whether the static routes are redistributed into BGP on the PE.
Redistribute_Static_IPV6	0	Flag to indicate whether the static routes are redistributed into BGP on the PE, if the Address family is IPv6
Redistributed_Protocol	1	List of routing protocols to be redistributed.
Rip_Metrics	0	Metric for redistribution associated with RIP.

Table 48-4 MPLS Repository Variables (continued)

Repository Variable	Dimension	Description
Routes_To_Reach_Other_Sites	2	List of one or more IP addresses to specify the static routes to put on the CE.
vrfName	0	Name of the VRF.

Table 48-5 provides a summary of the VPLS Repository variables available from Prime Fulfillment

Table 48-5 VPLS Repository Variables

Repository Variables	Dimension	Description
CARD_TYPE	0	Refers to NPE or UNI interface depending on whether the service is implemented with ethernet access.
VPLSBridgeDomainId	0	Bridge domain ID value.
VPLSCeEncapsulation	0	The encapsulation of the CE interface for a particular VPLS link.
VPLSCeInterfaceName	0	The name of the CE interface for a particular VPLS link.
VPLSCeMajorInterfaceName	0	The name of a major interface on a CE for a particular VPLS link.
VPLSCLECeFacingEncapsulation	0	The encapsulation of interfaces for a particular device facing the CE.
VPLSCLECeFacingInterfaceName	0	The interface name for a particular device facing the CE (the number can be more than 1 in case of a ring topology, hence any array).
VPLSCLEPeFacingEncapsulation	0	The encapsulation of interfaces for a particular device facing the PE
VPLSCLEPeFacingInterfaceName	1	The list of interface names for a particular device facing the PE (the number can be more than 1 in case of a ring topology, hence any array).
VPLSDisableCDP	0	The flag to specify if the CDP has been disabled on a UNI for a particular VPLS link.
VPLSFilterBPDU	0	The flag to specify whether the BPDUs will be filtered on a UNI for a particular VPLS link.
VPLSPeEncapsulation	0	The encapsulation of the PE interface for a particular VPLS link.
VPLSPeInterfaceDescription	0	The description assigned to the PE interface for a particular VPLS link.
VPLSPeInterfaceName	0	The name of the PE interface for a particular VPLS link.
VPLSPeMajorInterfaceName	0	The name of a major interface on a PE for a particular VPLS link.

Table 48-5 VPLS Repository Variables (continued)

Repository Variables	Dimension	Description
VPLSPeNeighbors	1	The list of PE POPs participating in a particular VPLS VPN.
VPLSPeVfiName	0	The VFI name assigned to a particular VPLS instance existing on the PE POP.
VPLSPeVlanId	0	The VLAN ID assigned to the PE for a particular VPLS link.
VPLSPeVpnId	0	The VPN ID assigned to a particular VPLS VPN.
VPLSSystemMTU	0	The maximum MTU value for a packet arriving on a UNI for a particular VPLS link.
VPLSTunnelCDPEnable	0	The flag to specify if the CDP packets will be tunneled to the remote site for a particular VPLS link.
VPLSTunnelCDPThreshold	0	The threshold value assigned for a CDP protocol before a violation action is reported on a UNI for a particular VPLS link.
VPLSTunnelRecoveryInterval	0	Interval for the UNI to recover from a shutdown scenario.
VPLSTunnelSTPEnable	0	The flag to specify if the STP packets will be tunneled to the remote site for a particular VPLS link.
VPLSTunnelSTPThreshold	0	The threshold value assigned for a STP protocol before a violation action is reported on a UNI for a particular VPLS link.
VPLSTunnelVTPEnable	0	The flag to specify if the VTP packets will be tunneled to the remote site for a particular VPLS link.
VPLSTunnelVTPThreshold	0	The threshold value assigned for a VTP protocol before a violation action is reported on a UNI for a particular VPLS link.
VPLSUniAging	0	The aging timer set on a UNI for a particular VPLS link.
VPLSUniDuplex	0	The duplex assigned to the UNI for a particular VPLS link.
VPLSUniMajorInterfaceName	0	The name of a major interface on a UNI device for a particular VPLS link.
VPLSUniMaxMacAddress	0	The maximum number of Mac addresses that can be learned on a UNI for a particular VPLS link.
VPLSUniPortSecurity	0	The port security option on a UNI for a particular VPLS link.

Table 48-5 VPLS Repository Variables (continued)

Repository Variables	Dimension	Description
VPLSUniProtocolTunneling	0	The flag to specify if the protocols will be tunneled to the remote site for a particular VPLS link.
VPLSUniSecureMacAddresses	1	The explicit list of Mac addresses that can be learned on a UNI for a particular VPLS link.
VPLSUniShutdown	0	The shutdown flag on a UNI for a particular VPLS link.
VPLSUniSpeed	0	The speed assigned to the UNI for a particular VPLS link.
VPLSUniViolationAction	0	The violation action option on a UNI for a particular VPLS link.
VPLSUseNativeVlan	0	The flag to specify if the native VLAN will be used on a UNI for a particular VPLS link.

Templates.

[Table 48-6](#) provides a summary of the VRF Repository variables available from Prime Fulfillment

Table 48-6 VRF Repository Variables

Repository Variable	Dimension	Description
Address_Family	0	Addressing scheme from Service Request.
Cerc_Hub_RT	0	Customer Edge Routing Community (CERC) for Hub Route Target.
Cerc_Spoke_RT	0	CERC for Spoke Route Target.
Export_Map	0	Name of the export map associated with the VRF.
Export_RT_List	0	One or more Route Targets (RTs) to be exported from the VRF.
Import_Map	0	Name of the import map associated with the VRF.
Import_RT_List	0	One or more RTs to be imported in the VRF.
Max_Routes	0	Maximum number of routes that can be imported into the VRF.
Max_Threshold	0	Percentage of the maximum number of routes that can be imported into the VRF.
PE	0	Name of the Provider Edge (PE) device.
PE_BGP_AS	0	BGP Autonomous ID for PE device.
RD	0	Route Distinguisher value for the VRF.
Vrf_Name	0	Name of the VRF.

Templates.

Table 48-7 provides a summary of the FlexUNI/EVC Repository variables available from Prime Fulfillment Templates.

Table 48-7 FlexUNI/EVC Repository Variables

Repository Variable	Dimension	Description
BACKUP_VC_ID	0	Backup virtual circuit ID for the AToM, where backup is configured for the primary pseudowire. This is applicable only for pseudowire core type connectivity between only two N-PEs.
CARD_TYPE	0	Refers to NPE or UNI interface depending on whether the service is implemented with ethernet access.
CONFIG_BRIDGE_DOMAIN	0	Value is true if USE_SVI is enabled.
CORE_TYPE	0	Core type connectivity. Possible values for this are: a) pseudowire, b) VPLS, c) Local connect.
EVC_LINK_ID	0	Returns top EVC link ID of EVC SR.
EVC_NPE_HOSTNAME	0	NPE device hostname in EVC SR.
EVC_SR_DESCRIPTION	0	EVC SR description.
EVC_SR_JOB_ID	0	SR JOB ID of EVC SR
FIEXUNI_ATM_VCD	0	Returns the ATM VCD/sub-interface value provided for ATM links.
FIEXUNI_ATM_VCI	0	Returns the ATM VCI value provided for ATM links.
FIEXUNI_ATM_VPI	0	Returns the ATM VPI value provided for ATM links.
FIEXUNI_REMOTE_HOSTNAME	0	Returns the remote peer's host name.
FIEXUNI_REMOTE_LOOPBACK	0	Returns the remote peer's loopback IP address.
FlexUNIBdName	0	Returns the Bridge Domain name used for IOSXR.
FlexUNIBgName	0	Returns the Bridge Group name used for IOSXR.
FlexUNIElineName	0	Returns the p2p Eline name used for IOSXR.
FlexUNIL2GroupName	0	Returns the L2VPN group name used for IOSXR.
FlexUNIPwClassName	0	Returns the PW class element name used for IOSXR.
IS_FLEX_UNI_LINK	0	Value is true if EVC LINK is FLEXUNI link.
LOCAL_CONNECT_NAME	0	Name of the connection between two Ethernet flow points (EFPs) using the connect command. Applicable only when there are two links that are FlexUNI/EVC enabled.

Table 48-7 FlexUNI/EVC Repository Variables (continued)

Repository Variable	Dimension	Description
MAC_ACL_RANGE	0	Range value specified for MAC ACL.
MATCH_INNER_VLANS	0	Contains the VLAN IDs that need to be matched for the ingress frame's inner VLAN tag. Applicable only for FlexUNI/EVC enabled links.
MATCH_OUTER_VLANS	0	Contains the VLAN IDs that need to be matched for the ingress frame's outer VLAN tag. Applicable only for FlexUNI/EVC enabled links.
PE_INTERFACE_NAME	0	N-PE interface of the link for a service. This is the same as the UNI_INTERFACE_NAME for direct connect links.
PE_OR_UNI_INTF_DESC	0	UNI interface description.
PUSH_INNER_VLAN_ID	0	Push a second Dot1q VLAN tag onto an ingress frame. Applicable only for links configured with FlexUNI/EVC.
PUSH_OUTER_VLAN_ID	0	Push a Dot1q VLAN (outer) tag onto an ingress frame. Applicable only for links configured with FlexUNI/EVC.
PW_TUNNEL_ID	0	Tunnel ID that is configured with a pseudowire class for the N-PE (applicable only for pseudowire core type selection).
SERVICE_INSTANCE_ID	0	Service instance ID (a number: 1 to 8000) corresponding to the EFP for a FlexUNI/EVC enabled link.
SERVICE_INSTANCE_NAME	0	Name of the EFP given to the Service instance being configured for a FlexUNI/EVC enabled link.
STD_UNI	0	Standard UNI status of the UNI interface.
STORM_CTL_BROADCAST_TRAFFIC	0	Storm control broadcast traffic value.
STORM_CTL_MULTICAST_TRAFFIC	0	Storm control multicase traffic value.
STORM_CTL_UNICAST_TRAFFIC	0	Storm control unicast traffic value.
SYSTEM_MTU	0	System MTU size used.
TRANSLATE_INNER_VLAN_ID	0	Target inner VLAN ID of a frame that is being translated (VLAN translation). Applicable only for FlexUNI/EVC enabled links. This is applicable for 1:2/2:2 types of translation.
TRANSLATE_OUTER_VLAN_ID	0	Target outer VLAN ID of a frame that is being translated (VLAN translation). Applicable only for FlexUNI/EVC enabled links. This is applicable for any kind of translations (1:1/1:2/2:1/2:2).

Table 48-7 *FlexUNI/EVC Repository Variables (continued)*

Repository Variable	Dimension	Description
TUNNEL_CDP_DROP_THRESHOLD	0	CDP DROP threshold value used.
TUNNEL_STP_DROP_THRESHOLD	0	STP DROP threshold value used.
TUNNEL_VTP_DROP_THRESHOLD	0	VTP DROP threshold value used.
UNI_AGING	0	The aging value of the UNI.
UNI_ENCAPSULATION_TYPE	0	Encapsulation on the UNI. Possible values are: a) Dot1Q Trunk, b) Dot1Q Tunnel, c) Access.
UNI_INTERFACE_NAME	0	UNI of the link for a service. This is the same as PE_INTERFACE_NAME for direct connect links.
UNI_PORT_SECURITY	0	The port security status of the UNI.
UNI_SHUTDOWN	0	The UNI shutdown status.
UNI_SPEED	0	The speed value of the UNI.
UNI_VIOLATION_ACTION	0	Type of violation action used.
UPE_FACING_INTERFACE_NAME	1	Arrays of one or two elements, containing names of NNI interfaces on NPE towards the U-PE. Two interfaces exist if access is via a ring, otherwise just one is present.
USE_SPLIT_HORIZON	0	Value is true if split horizon is enabled.
USER_DEFINED_ACL_NAME	0	User defined ACL name used in the attachment circuit.
VC_ID	0	The virtual circuit ID for the AToM where pseudowire is the core connectivity type between two N-PEs.
VLAN_ID	0	VLAN ID corresponding to the service on PE devices for the link. For links that are configured with FlexUNI/EVC, this is applicable only on N-PE, while MATCH_OUTER_VLANS represents the service for that link.
VLAN_NAME	0	VLAN name configured for the VLAN ID corresponding to the link for the service.
VPLS_VPN_ID	0	VPLS VPN ID for VPLS core type connectivity.
VPN_ID	0	VPN name associated to EVC SR.

Importing and Exporting Templates

The **importExportTemplateDB** tool is available to import and export templates into and from an Prime Fulfillment database.



Note If a **Negate** template is present, it is automatically imported or exported for every import or export template.

You can import or export the complete or partial template database by specifying appropriate arguments. You can find this tool at: **\$PRIMEF_HOME/bin/importExportTemplateDB.sh**.

Enter the following:

importExportTemplateDB.sh *<admin_user_id>* *<password>* [*<other_arguments>*]

where:

<admin_user_id> is user identifier for someone with the **admin** role.

<password> is the password for the one with the **admin** role.

<other_arguments> is any combination of the following arguments separated by a space:

-nooverwrite

If you choose to use this **nooverwrite** argument, to prevent the overwriting of existing templates in the database, it must precede all other arguments and must be in the third position after *<admin_user_id>* and *<password>*.



Note The default (when **nooverwrite** is not specified) is to overwrite the templates.

-exp_db *<dest-dir>*

Use this argument to export all templates and datafiles in the database, where *<dest-dir>* is the destination directory to which you want to export.

-imp_db *<src-dir>*

Use this argument to import all the files in *<src-dir>* into the database, where *<src-dir>* is the source directory from which you want to import. The files in *<src-dir>* are created by the **exp_db** process.

-exp_template_folder *<src-folder-path>* *<dest-dir>*

Use this argument to export a database template folder and its subfolders, where *<src-folder-path>* is the full path of the template folder to export and *<dest-dir>* is the directory where to place the exported files.

-imp_template_folder *<src-dir>* *<dest-folder>*

Use this argument to import all files in *<src-dir>* into the database, where *<src-dir>* is the source directory to import, and *<dest-folder>* is the destination import template folder.

-imp_template *<srcfile>* *<dest-folder>* *<template-name>*

Use this argument to import a template into the database, where *<srcfile>* is the full path of the template to import, *<dest-folder>* is the full path of the parent folder, and *<template-name>* is the template name in the database.

-imp_datafile *<srcfile>* *<dest-template>* *<datafile-name>*

Use this argument to import a template datafile into the database, where *<srcfile>* is the full path of the datafile to import, *<dest-template>* is the full path of the parent template, and *<datafile-name>* is the datafile name in the database.

-exp_template *<template-pathname>* *<output-file>*

Use this argument to export the database template to a file, where *<template-pathname>* is the full path of the template to export, and *<output-file>* is the output filename.

-exp_datafile *<datafile-pathname> <output-file>*

Use this argument to export a template datafile to a file, where *<datafile-pathname>* is the full path of the template datafile to export, and *<output-file>* is the output filename.

Template Usage

The following questions and answers can help you troubleshoot. The following topics are for Template Manager, which is explained in [Chapter 48, “Overview of Templates and Data Files”](#):

- [How do I split a string?, page 48-40](#)
- [How do I obtain address information from the given IP address?, page 48-41](#)
- [How do I obtain the octets from the given IP address?, page 48-41](#)
- [How do I call a subtemplate in a template?, page 48-41](#)
- [How do I concatenate two strings?, page 48-41](#)
- [How can I convert a string to an integer and how can I increase the last octet of the IP address by one?, page 48-42](#)
- [Can I use nested if statements?, page 48-42](#)
- [How can I perform basic arithmetic operations?, page 48-42](#)
- [How can I retrieve data from a two-dimensional array and what is the use of \\$velocityCount?, page 48-43](#)
- [How can I print \\$a instead of its value?, page 48-43](#)
- [What is the difference between #include\(\) and #parse\(\)?, page 48-44](#)
- [What is a macro and how is it used?, page 48-45](#)
- [What is a range operator and how can I use it?, page 48-45](#)
- [How can I split strings containing special characters?, page 48-46](#)
- [How can I use repository variables?, page 48-46](#)
- [How can I use a variable as a dynamic URL?, page 48-46](#)
- [Can I see more examples?, page 48-46](#)

How do I split a string?

Prime Fulfillment provides a function **substringToDelim()**, which can split the given string and return the substring based on the given delimiter.

Syntax:

substringToDelim (srcString, delimChar, 0/1)

where:

0 returns the string before the delimiter.

1 returns the string after the delimiter.

Usage: **\$b=\$TMSystem.substringToDelim("10.11.230.145", ".230.145", "0")**

Result: The value of **\$b** is **10.11**. If **1** is specified instead of **0**, the value of **\$b** is **230.145**.

How do I obtain address information from the given IP address?

Prime Fulfillment provides the functions that can be used to get the address, mask, and reverse mask from the given IP address.

Usage:

```
$TMSsystem.getAddr ("10.33.4.5/30") returns 10.33.4.5
$TMSsystem.getMask ("10.33.4.5/30") returns 255.255.255.252
$TMSsystem.getReverseMask ("10.33.4.5/30") returns 0.0.0.3
$TMSsystem.getNetworkAddr ("10.33.4.5/30") returns 10.33.4.4
$TMSsystem.GetClassfulNetworkAddr ("10.33.4.5/30") returns 10.0.0.0
$TMSsystem.CurrentTimeInIOSFormat () returns hh:mm:ss day_of_month month_of_year year
```

How do I obtain the octets from the given IP address?

Prime Fulfillment provides the functions that can return the octets when called.

Usage:

```
$TMSsystem.getOctet1($ipAddr) returns the first octet of ipAddr
$TMSsystem.getOctet2($ipAddr) returns the second octet of ipAddr
$TMSsystem.getOctet3($ipAddr) returns the third octet of ipAddr
$TMSsystem.getOctet4($ipAddr) returns the fourth octet of ipAddr
```

How do I call a subtemplate in a template?

A subtemplate can be called in a main template. The subtemplate being called should be called with its datafile. The variable is declared as a subtemplate. The location of the subtemplate is specified in the datafile.

Usage: In the template body the subtemplate is declared as:

```
$a. callWithDatafile("data1")
```

where:

the variable **a** is declared as a subtemplate in the variables

data1 is the name of the datafile of the subtemplate, and

in the datafile the path of the subtemplate path is specified.

How do I concatenate two strings?

Concatenation of strings is simple.

For example:

where: **\$a=vpnsc** and **\$b=properties**

then: **\${a}\${b}** concatenates these two strings and gives the result as **vpnscproperties**.

or, **\${a}_\${b}** gives the result as **vpnsc_properties**.

How can I convert a string to an integer and how can I increase the last octet of the IP address by one?

The last octet of the IP address can be increased by using the following code:

```
#set($d=$TMSsystem.getOctet1($c))
#set($e=$TMSsystem.getOctet2($c))
#set($f=$TMSsystem.getOctet3($c))
#set($g=$TMSsystem.getOctet4($c))
#set($valueOfString = $g)
#set($valueOfCharsCount = $valueOfString.length() - 1)
#set($valueOfVector = "0123456789")
#set($valueOfBase = 1)
#set($valueOfInt = 0)
#foreach($valueOfCharIterator in $valueOfCharsCount..0)
#set($valueOfChar=$valueOfString.charAt($valueOfCharIterator).toString())
#set($valueOfInt = $valueOfInt + $valueOfVector.indexOf($valueOfChar) * $valueOfBase)
#set($valueOfBase = $valueOfBase * 10)
#end
#set($valueOfInt = $valueOfInt+1)
```

The incremental value is `$d.$e.$f.$valueOfInt`

Can I use nested if statements?

If statements can be nested. Proper care must be taken for indentation when nesting **if** statements. The following code shows the usage of nested **if** statements, **elseif** statements, and the comparisons made in the **if** clause.

```
#if($a=="a") // here: string comparison is made
--
  #if($b || $d) // here: $b and $d are the Boolean expressions. || equals OR and && equals AND
  --
    #if(!$c) // here: $c can be integer, string, or Boolean.
    ---
      #if($p<10)// here: $p is a integer.
      #elseif($p==10)
      #end
    #end
  #end
#end
#end
```

How can I perform basic arithmetic operations?

Velocity Template Language (VTL) supports built-in mathematical functions that can be used in the templates with the set directives.

Usage:

```
#set($a = $b + 3)
#set($a = $b - 6)
#set($a = $b * 6)
#set($a = $b / 5)
#set($a = $b % 2)
```

**Note**

Only integers are valid for performing mathematical operations in the VTL.

How can I retrieve data from a two-dimensional array and what is the use of \$velocityCount?

The default name for the loop counter variable reference, which is specified in the velocity.properties file, is **\$velocityCount**. By default the counter starts at **1**, but this can be set to either **0** or **1** in the velocity.properties file at:

\$PRIMEF_HOME/resources/webserver/tomcat/shared/lib/velocity-dep-VelocityVersion.jar (where the current *VelocityVersion* is 1.3.1-rc2). The associated settings are:

```
directive.foreach.counter.name=velocityCount
directive.foreach.counter.initial.value=1
```

Data from an array can be obtained by using **get(\$i)**

where: **\$i** is the \$velocityCount.

The following example illustrates the usage of the method **get()**:

```
Usage: #foreach ($Acl in $ACL-List)
      #set ($i = $velocityCount)
      #foreach ($protocol in $Protocol-Lists.get($i))
      #set ($j = $velocityCount)
      access-list $Acl permit $protocol $Source-IP.get($i).get($j)
      #end
      #end
```

where:

\$ACL-List is a one-dimensional array.

\$Protocol-Lists and **\$Source-IP** are two-dimensional arrays.

Here the \$velocityCount is set to **1** by default. It can be changed in velocity.properties, if desired.

How can I print \$a instead of its value?

Printing a value without processing is done by use of the character ****, even if the value of the variable for **a** is defined.

Usage:

\\$a gives output as **\$a** if **\$a** is defined. If **\$a** is not defined, it is printed as **\\$a**.

What is the difference between `#include()` and `#parse()`?

The `#include("velocity.txt")` directive allows you to import a file and then include the file in the location where it is defined. The content of the file is made available to the template engine. The *.vm files can also be called by using `#include`. The name of the file can also be passed by a variable. For security reasons, the file should be included under **TEMPLATE_ROOT** (/vob/ntg/dev/resources/templatesystem).

The `#parse("velocity.vm")` directive allows you to import a local file that contains VTL. Velocity will parse the VTL and render the template specified. The template that `#parse` references must be included under **TEMPLATE_ROOT**. The `#parse` directive only takes a single argument. VTL templates can have `#parse` statements referring to templates that in turn have `#parse` statements. The default value of the **directive.parse.max.depth** property is set to 10, in the **velocity.properties** file at: **\$PRIMEF_HOME/resources/webserver/tomcat/shared/lib/velocity-dep-VelocityVersion.jar** (where the current *VelocityVersion* is 1.3.1-rc2) and can be modified, if desired.



Note

If the **directive.parse.max.depth** property is not present in the **velocity.properties** file, the default is set to 10.

Example:

In **TEMPLATE_ROOT**, the file **velocity.vm** has the following content:

```
welcome to the parse file
The count is $count
#set($count = $count - 1)
#set($cl-list="cl1","cl2","cl3")
#foreach($i in $cl-list)
ipcommunity-list permit $i 30:20
#end
The count is $count
returning from parse
```

The template body contains the following:

```
#set($count=8)
#include("velocity.vm")
-----
#parse("velocity.vm")
-----
```

```
welcome back to template
The value of count is $count
```

The following O/P is obtained:

```
welcome to the parse file
The count is $count
#set($count = $count - 1)
#set($cl-list="cl1","cl2","cl3")
#foreach($i in $cl-list)
ipcommunity-list permit $i 30:20
#end
The count is $count
returning from parse
```

```

-----
welcome to the parse file
The count is 8
ipcommunity-list permit cl1 30:20
ipcommunity-list permit cl2 30:20
ipcommunity-list permit cl3 30:20
The count is 7
returning from parse
-----
welcome back to template
The value of count is 7.

```

**Note**

The previous examples clearly show that variables are parsed in the **#parse** directive and not in the **#include** directive.

What is a macro and how is it used?

The directive macro is almost similar to a function. This has a set of statements, which can be called repetitively.

Example:

```

#macro(community $CL $bgp-list)
  #foreach($bgp in $bgp-list)
    ip $CL standard permit $bgp
  #end
#end

#set($bgp_list = "20:10","30:10","40:10","50:10")
#set($CL = "community-list")

#community($CL $bgp_list)

```

Here, the macro name of **community** is defined. The macro takes two arguments **\$CL** and **\$bgp-list**. The macro is called at the end line.

The output of the previous template is:

```

ip community-list standard permit 20:10
ip community-list standard permit 30:10
ip community-list standard permit 40:10
ip community-list standard permit 50:10

```

What is a range operator and how can I use it?

The range operator can be used in conjunction with **#set** and **#foreach** statements. It is used to produce an object array containing integers. The range operator has the following construction **n..m**.

Example:

```
#set($a=0..2)
#foreach($b in $a)
    $b
#end
#foreach($c in -2..2)
    $c
#end
```

How can I split strings containing special characters?

```
#foreach ($i in $PE_Intf_Name.split('\.')) $i #end
```

here: In the first iteration, **\$i** contains the string before the period, and in the second iteration, **\$i** contains the string after the period.

How can I use repository variables?

Repository variables can be selected in the datafile. When a template along with a datafile is associated with a Service Request and the Service Request is deployed, then the value of the repository variable gets substituted.

How can I use a variable as a dynamic URL?

A variable declared as a dynamic URL can call the URL, by the method:

```
callUrl(String S)
```

For example: **\$a.callUrl("http://www.cisco.com")**

Can I see more examples?

Examples are given for:

- [Usage of Strings, page 48-47](#)
- [Usage of a Macro, page 48-48](#)
- [Usage of Subtemplates, page 48-49](#)

Usage of Strings

The body of the template contains:

This example illustrates the usage of strings

```
#set($a="Fast")
#set($b="ethernet")
interface ${a}_${b}

#foreach ($i in $PE_Intf_Name.split('\.'))
$i
#end

#set($c="10.11.230.145")
#set($b=$TMSsystem.substringToDelim($c, ".230.145", "0"))
interface Loopback1
description By VPN-SC
ip vrf forwarding V31:eigrpfm
ip address ${b}.20.34 255.255.255.255
no ip directed-broadcast

#set($b=$TMSsystem.substringToDelim($c, ".230.145", "1"))
interface Loopback1
description By VPN-SC
ip vrf forwarding V31:eigrpfm
ip address 20.45.${b} 255.255.255.255
no ip directed-broadcast

#set($c="10.33.4.5/30")
#set($d=$TMSsystem.getAddr($c))
The Address of $c is $d
#set($d=$TMSsystem.getMask($c))
The mask of $c is $d
#set($d=$TMSsystem.getReverseMask($c))
The Reverse mask of $c is $d
#set($d=$TMSsystem.getNetworkAddr($c))
The network address of $c is $d

#set($e=$TMSsystem.currentTimeInIOSFormat())
The current time in IOS format is : $e

-----
getting the octets from the ipaddress
#set($c="10.33.4.5")
#set($e=$TMSsystem.getOctet1($c))
The first Octet of $c is $e
#set($e=$TMSsystem.getOctet2($c))
The second Octet of $c is $e
#set($e=$TMSsystem.getOctet3($c))
The third Octet of $c is $e
#set($e=$TMSsystem.getOctet4($c))
The fourth Octet of $c is $e
```

The variables are declared as strings, integers, or sub-templates accordingly.

The Output of the above template body is:

```
interface Fast_ethernet
```

```
10
11
12
13
```

```
interface Loopback1
description By VPN-SC
ip vrf forwarding V31:eigrpfm
ip address 10.11.20.34 255.255.255.255
no ip directed-broadcast
```

```
interface Loopback1
description By VPN-SC
ip vrf forwarding V31:eigrpfm
ip address 20.45.230.145 255.255.255.255
no ip directed-broadcast
```

```
The Address of 10.33.4.5/30 is 10.33.4.5
The mask of 10.33.4.5/30 is 255.255.255.252
The Reverse mask of 10.33.4.5/30 is 0.0.0.3
The network address of 10.33.4.5/30 is 10.33.4.4
```

```
The current time in IOS format is: 00:17:01 21 Aug 2006
```

```
-----
getting the octets from the ipaddress
The first Octet of 10.33.4.5 is 10
The second Octet of 10.33.4.5 is 33
The third Octet of 10.33.4.5 is 4
The fourth Octet of 10.33.4.5 is 5
```

Usage of a Macro

The body of the template contains:

```
## This example illustrates the usage of macro
```

```
#macro(community $CL $bgp-list)
#foreach($bgp in $bgp-list)
ip $CL standard permit $bgp
#end
#end
```

```
#set($bgp_list = "20:10","30:10","40:10","50:10")
#set($CL = "community-list")
```



```
#community($CL $bgp_list)
```

The Output is obtained as:

```
ip community-list standard permit 20:10
ip community-list standard permit 30:10
ip community-list standard permit 40:10
ip community-list standard permit 50:10
```

Usage of Subtemplates

The body of the template is as follows:

This example illustrates the usage of the sub-template

```
$a.callWithDatafile("data1")
```

Template Editor

Template Information	
Template Name *	<input type="text"/>
Description:	<input type="text"/>
Body *	<div style="border: 1px solid black; height: 150px;"></div>
Has Negate Template	<input type="checkbox"/>
Has User Reference	<input type="checkbox"/>

Note: * - Required Field

238387

The variable **a** is declared as a subtemplate. The datafile provided here, **data**, must be a datafile for the template **a**, which must also exist. In the datafile of the main template, the path of the subtemplate is specified.

In the datafile of the main template, the specified path of the subtemplate might be the same directory or a different directory.

Can I see more examples?