Adding Additional Information to Services

This appendix describes how the additional information feature is supported in Prime Provisioning.

Note For MPLS and EVC services, it is recommended that you use a new policy customization feature. For more information, see Chapter 7, "Customizing EVC and MPLS Policies".

It contains the following sections:

Overview

The additional information feature allows a set of attributes (name/value pairs) to be defined in an XML file by the user. The file is subsequently associated with a policy. The additional information attributes define values to be associated with a service request. They define labels and appearance in the GUI. In the service request workflow, these values can be entered by the user. It is also possible to access these additional attribute values either from templates or from the xDE provisioning logic, to provide data values that will be configured as part of a service. Using additional attributes in combination with templates allows template attribute values to be prompted for in the policy and service request GUI, instead of having to create data files with these values. This appendix provides the information needed to understand and use the additional information feature in Prime Provisioning.

Prerequisites and Limitations

Be aware of the following prerequisites and limitations of the additional information feature:

- The additional information feature is only supported for MPLS, L2VPN, VPLS, and EVC services.
- MPLS-TP and TEM policies and service requests do not support additional information.
- VRF services requests do not have policies and so do not support additional information.
- Before using this feature in a supported policy or service request type, you must create an additional information definition file. This is an XML file that defines the user-defined attribute/value pairs. You later load this definition file in a step within the policy workflow. For more information about this, see Creating the Additional Information Definition File, page E-7.

Summary of the Additional Information GUI Workflow

The following steps summarize the tasks you need to perform to implement additional information in Prime Provisioning. The remaining sections in this appendix provide detailed information on these topics.

1. Create an additional information definition file (perhaps using the supplied XSD to validate). This file defines the additional information attributes.
2. Create a template that refers to the additional attribute values or, alternatively, extend the xDE provisioning logic.
3. Create a single default data file for the template.
4. Optionally add the negate template and negate data file.
5. Create a policy of the appropriate policy type.
6. Go to the Additional Information window in the policy creation workflow.
7. Load in the additional information definition file that was created. The file will be parsed and validated, and any errors displayed in the GUI.
8. Fill in the values in the provided fields, if needed. You can define these in the additional information definition file if these are standard values that do not need to be changed.
9. In the policy workflow, mark the additional information attributes as editable or not. This determines whether or not you can edit these values in the service request based on the policy.
10. In the policy workflow, enable templates and reference the templates that access the additional values.
11. Save the policy. The additional information will be parsed and validated, and any errors displayed in the GUI.
12. Create a service request based on the policy.
13. The Service Request Editor window in the service request workflow will display the additional information attributes and allow you to edit them (if they are editable).
14. Save the service request. The additional information will be parsed and validated, and any errors displayed in the GUI.

Setting Additional Information in the Policy Workflow

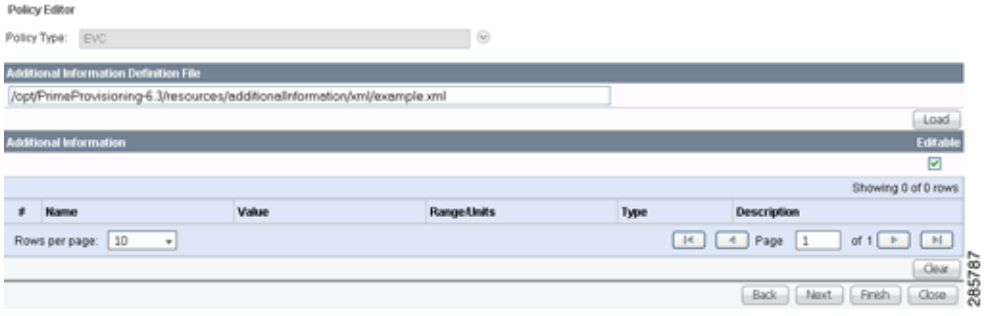
Perform the following steps to use the additional information feature within the supported policy types.

Step 1 Edit or create a supported policy type for which you want to add additional attributes.

Step 2 Navigate through the policy workflow windows and set attribute values as required for your configuration.

Several windows into the workflow, an Additional Information window appears. like the one shown in Figure E-1. This window looks and functions the same in all of the policy types that support the additional information feature.

Figure E-1 Additional Information Window



This window is the second to the last window of the policy workflow, and it appears before the Template Association window.

Use of the Additional Information window is optional.

Step 3 Click the **Load** button to load the XML definition file that defines the attribute/value pairs for the additional information to be added.

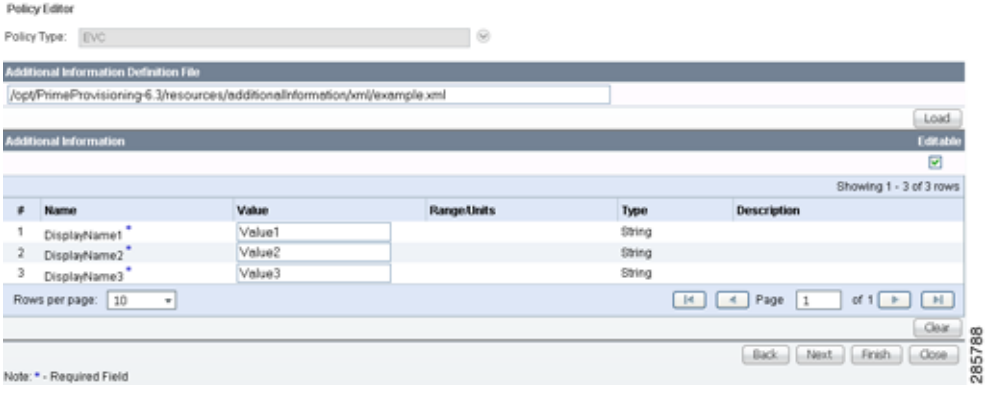
Note For information on how to create this file, see Creating the Additional Information Definition File, page E-7.

The default path and name of the definition file is:

$PRIMEP_HOME/resources/additionalInformation/xml/example.xml

The window refreshes and the attribute/value pairs from the definition file appear in the Additional Information section, as shown in Figure E-2.

Figure E-2 Attributes Loaded from an External XML Definition File



Step 4 If desired, you can click the Clear button to clear the attributes shown in the Display Section of the window.

Step 5 Check or uncheck the **Editable** check box to set all of the Additional Information attributes as editable or not.

You cannot make individual attributes editable or not.

Step 6 Set the values for the additional information attributes as desired for the policy.

See the discussion below for comments about the contents and behavior of this section of the window.

Step 7 Click **Next** to proceed to the next step of the policy workflow.

Step 8 Complete the policy workflow following the standard steps in Prime Provisioning.

Be aware of the following points concerning the contents and behavior of the Additional Information section of the window:

- Additional Information attributes are grouped together in the GUI based on how they are defined in the additional information definition file.
- If groups are defined, then for each group the group name is displayed above a paging table containing the additional information attributes.
- If no groups are defined in the definition file, then only a paging table containing the additional information attributes is displayed.
- Each attribute is displayed in a row in the paging table.
- The Name column contains the DisplayName of the attribute, as defined in the definition file. If an attribute is marked as required in the definition file, a superscript asterisk is appended to the DisplayName. This does not indicate that the attribute must have a value in the policy, but that this is how it is defined in the definition file and that a value will be required for this attribute in a service request using this policy.
- The Value column contains the Value of the attribute, as defined in the definition file.
- The Range/Units column contains a combination of the range and units for the attribute.
- The Description column contains the Description of the attribute, as defined in the definition file.

Validation Checks Done to the Definition File in the Policy Workflow

In additional to the XSD validation, the parsing checks performed, and the validation performed for the additional information definition file, the following further validation checks are performed when a policy with additional information is saved to the Prime Provisioning database. If the Additional Information section is marked as not editable (that is, the Editable check box is left unchecked), then any attributes marked as required need to have a value defined. A validation error is generated if this is not the case. This restriction is due to the fact that in a service request based on the policy, all required Additional Information attributes must have a value. So if you cannot edit the value (because the Additional Information is not editable) then you will never be able to create a service request based on the policy.

For more information about validation checks done on the additional information in the policy workflows, see How the XSD is Validated, page E-10.

Setting Additional Information in the Service Request Workflow

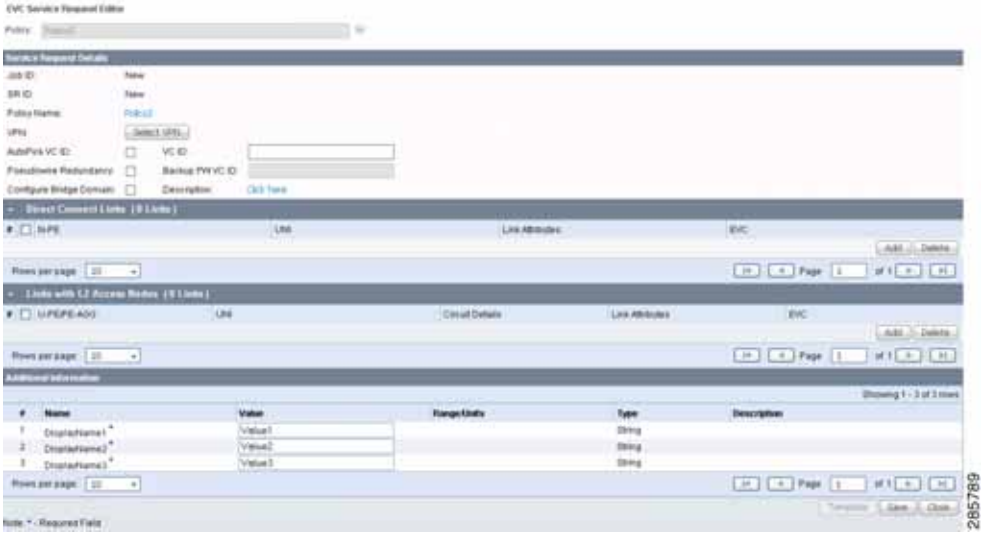
Perform the following steps to use the additional information feature in the service request workflow.

Step 1 Create or edit a service request based on a policy which was created using the additional information feature.

Step 2 Navigate to the Service Request Editor window within the service request workflow.

If the policy on which the service request is based had Additional Information attributes defined, these attributes are displayed, as shown in Figure E-3.

Figure E-3 Additional Information Attributes in the EVC Service Request Window



The attributes are displayed below any existing attributes of the Service Request Editor window. The format of the attribute in the Attribute Information is the same as the corresponding section in the policy.

Step 3 Set the attributes within the Service Request Editor based on the requirements for your configuration.

Be aware of the following points concerning the attributes Additional Information section:

- If the Additional Information attributes are editable, the values of the attributes can be changed.
- If the Additional Information attributes are not editable, the values are greyed-out and so cannot be changed.
- If there are no Additional Information attributes in the policy that the service is based upon, the Service Request Editor window will not show the Additional Information section.
- You must set values for attributes marked as required. If this is not done, a validation error is generated when you attempt to save the service request.

Step 4 At this stage, you may also add templates to the devices in order to map template variables to additional information attributes. For more information about this, see Using Additional Attributes with Templates and Data Files, page E-5.

Step 5 Click **Save** to save the service request.

Using Additional Attributes with Templates and Data Files

You can map template variables to Additional Information attributes in two places in Prime Provisioning:

- When a template is created. To do this, perform the following steps:
  1. Edit the template variables that you want to map, and define them as type String.
  2. Enter the Additional Information attribute name as the default value for the template variable. You must use the exact name that is defined in the additional information definition file.

Note The attribute used in the template must start with $ (for example, *$name*), as this indicates that this value will be substituted with another value at deployment time. When you create the default value or the data file for this attribute, then you give the exact name of the Additional Information attribute. The Additional Information attribute name must start with a $, as this indicates to the Template Manager that this attribute will be substituted with the actual value and is not just a hard-wired string.

- When a template data file is created. To do this, enter the Additional Information attribute name as the value for the template variable. You must use the exact name that is defined in the additional information definition file.

After you have performed either of these approaches, then when you associate a template and/or template data file with a policy or service request, the template variables are substituted with the values of the corresponding Additional Information attributes defined by the user in the policy or the service request.

Using Additional Attributes with xDE Provisioning

Additional Information attributes are added to the XML document that is passed to the xDE provisioning engine, and thus they can be accessed by any of the xDE procedures.

Append the following XML block to this XML document:

```
<additionalInformation>
<attribute>
<name>Name1</name>
<value>123</value>
</attribute>
</additionalInformation>
```

Note The attribute XML block must be repeated for each *additionalInformation* attribute.

In the current xDE procedures for provisioning, the request attribute is passed to every procedure that contains the input XML file. To use *additionalInformation* attribute values in the xDE procedure, you can extract the value of attribute Name1 from the MPLS SR XML request doc as follows:

```
xml.xpathreference($serviceRequest,
"/MplsSR/additionalInformation/attribute[name=\"Name1\"]/value/text()")
```

Alternatively, you can access the additional information attribute values via the $additionalInformation attribute that is passed to all xDE procedures. This attribute contains a map of all the additional information attribute name/value pairs. For example:

```
map.get($additionalInformation, "Name1")
```

returns the value associated with the Name1 attribute

Creating the Additional Information Definition File

This section provides reference information you can use to create an additional information definition file. This is an XML file containing a minimum set of mandatory XML elements, plus additional optional elements. This file is later loaded into a policy as described in the section Setting Additional Information in the Policy Workflow, page E-2.

Minimum Mandatory XML Elements

Example E-1 is an example additional information definition file that contains the minimum information needed to define an additional information attribute.

Example E-1 Additional Information Definition File with Minimum XML Elements

```
<additionalInformation>
<attribute>
        <name>Name1</name>
        <value>Value1</value>
    </attribute>
</additionalInformation>
```

Explanation of the mandatory XML elements:

- *additionalInformation*—The *additionalInformation* block starts and ends the definition file.
- *attribute*—The *attribute* block can be repeated for as many attributes as you would like to define. There must be only one *name* element and one *value* element in each *attribute* block.
- *name*—The *name* element must have non-null value, and this value must be unique with respect to the values of other *name* elements in the additional information definition file.
- *value*—The *value* element can have any value (including null), and this value does not need to be unique with respect to the values of other *value* elements in the additional information definition file.

Optional XML Elements

The additional information defintion file also may contains optional XML elements . This section describes the following optional elements:

- *group*
- *attribute/displayName*
- *attribute/description*
- *attribute/required*
- *attribute/type*
- *attribute/type/string*
- *attrribute/type/integer*
- *attribute/type/ipv4Address*
- *attribute/type/ipv6Address*
- *attribute/type/enumeration*

Information is provided on how each element is parsed and what conditions generate errors.

For an example additional information definition file that contains some of the optional elements that can be configured, see Example of the Additional Information Feature, page E-10.

group

There can be zero or more *group* elements.

Each *group* must have at least 1 attribute block. Zero attributes in a *group* will generate an error when the file is loaded.

If there is a *group* defined, then you cannot define attributes at the same level (that is, outside a *group*). *groups* and *attributes* at the same level will generate a parsing error when the file is loaded.

group elements must have a *name*, but *name* can be blank.

A *group name* must be unique, including blank names (that is, you can only have 1 blank *name*). A non-unique *group name* will generate a duplicate name error when the file is loaded.

attribute/displayName

The *displayName* element contains the text that is displayed in the Name column of the attribute in the the Additional Information table in the policy and service request workflow.

If *displayName* is not defined, it defaults to the text in the *name* element.

attribute/description

The *description* element contains the text that is displayed in the Description column of the attribute in the Additional Information table in the Policy and Service Request workflow. If *description* is not defined, it defaults to an empty string.

attribute/required

The *required* element contains a Boolean that indicates whether or not the attribute is required. If set to true, then a superscript asterisk is placed beside the *name* text that is displayed in the Name column of the attribute in the Additional Information table in the policy and service request workflow.

For policies, if an attribute is set as required, then it only needs to have a value if the Additional Information is set as not editable. Otherwise, the attribute does not need to have a value.

For service requests, if an attribute is set as required, then it needs to have a value set.

If *required* is not defined, it defaults to true.

attribute/type

The *type* element describes what type of attribute is being defined.

The available types are:

- *string*
- *integer*
- *ipv4Address*

- *ipv6Address*
- *enumeration*

If no *type* element is defined, then the default type is *string* (no ranges or regex is defined).

If the *type* element is defined but does not have one of the available types as a sub-element (either no type or a non-supported type), then this will generate a parsing error when the file is loaded.

If there are more than one *type* elements for an attribute, then a parsing error will be generated when the file is loaded.

attribute/type/string

The *string* type has a number of optional parameters that describe the range and units, as follows:

- *minLength*—Defines the minimum length of the string. The attribute string value length must be greater than or equal to this in order to pass validation. If *minLength* is not defined, then the default is 1.
- *maxLength*—Defines the maximum length of the string. The attribute string value length must be less than or equal to this in order to pass validation.
- *rangeUnits*—Defines the units to be displayed in the Range/Units column, in conjunction with the range parameters if defined. If *rangeUnits* is not defined then the default is “characters”.
- *regex*—Defines a regex that will be used to validate the attribute string value. The string value must satisfy the regex to pass validation. In addition, if *regex* is defined, then the *rangeDescription* will be appended with “Pattern: regex”.

attribute/type/integer

The *integer* type has a number of optional parameters that describe the range and units, as follows:

- *lower* —Defines the lower value of the range. The attribute *integer* value must be greater than or equal to this to pass validation.
- *upper* —Defines the upper value of the range. The attribute *integer* value must be less than or equal to this to pass validation.
- *rangeUnits*—Defines the units to be displayed in the Range/Units column, in conjunction with the range parameters if defined. If *rangeUnits* is not defined, then the default is an empty string.

attribute/type/ipv4Address

The *ipv4Address* type has a number of optional parameters that describe the range and units, as follows:

- *ipv4Lower*—Defines the *ipv4Lower* value of the range. The attribute *ipv4Address* value must be greater than or equal to this to pass validation.
- *ipv4Upper*—Defines the *ipv4Upper* value of the range. The attribute *ipv4Address* value must be less than or equal to this to pass validation.

attribute/type/ipv6Address

The *ipv6Address* type has a number of optional parameters that describe the range and units as follows:

- *ipv6Lower*—Defines the *ipv6Lower* value of the range. The attribute *ipv6Address* value must be greater than or equal to this to pass validation.

- *ipv6Upper*—Defines the *ipv6Upper* value of the range. The attribute *ipv6Address* value must be less than or equal to this to pass validation.
- *rangeUnits*—Defines the units to be displayed in the Range/Units column, in conjunction with the range parameters, if defined. If *rangeUnits* is not defined, then the default is an empty string.

attribute/type/enumeration

The *enumeration* type has a number of optional parameters that describe the range and units as follows:

- *enumOptions*—Defines the enumeration options for the attribute.
  - 1 or more *enumOptions* elements can be defined.
  - If there is not at least 1 *enumOption* element defined, then a parsing error will be generated when the file is loaded.
  - An empty string is not a valid *enumOption* value. If any of the *enumOption* elements have empty strings, a parsing error will be generated when the file is loaded.
- *rangeUnits*—Defines the units to be displayed in the Range/Units column, in conjunction with the range parameters, if defined. If *rangeUnits* is not defined, then the default is an empty string.

How the XSD is Validated

The additional information XML is validated using the XML schema definition (XSD). The XSD is defined in the main JAR file and so cannot be edited by the user. However, a copy of the file is available in the following location for users wanting to build additional information definition files:

$PRIMEP_HOME/resources/additionalInformation/extAttrs.xs

There is a DCPL property that allows the user to turn on/off the XSD validation. The DCPL property is **additionalInformation.XML.validateWithXSD**. It is on by default.

How the Additional Information Definition File is Validated

In addition to the XSD validation and the parsing checks performed, the following further validation checks are performed on the additional information definition file when it is loaded into a policy:

- *Enumeration* type—If an attribute value is defined but does not match one of the *enumeration* options, then a validation error is generated. If there are duplicate *enumeration* options, then a validation error is generated.
- *integer*, *ipv4Address* and *ipv6Address* types— If an attribute value is defined, then it is checked against the range (if no range defined then the defaults are used) and a validation error is generated if it is outside this range.
- *string* type—If an attribute value is defined, then in addition to the range checks (mentioned above), it must also match the *regex* (if it has been defined).

Example of the Additional Information Feature

This section provides an end-to-end example of the additional information feature. The example provides the following information:

- Template
- Template data file
- Additional information definition file
- List of attributes that display in the GUI
- Example GUI input and generated configlets

Template

Here is the example policy template body. The template is very generic. It shows an E-line service for an access port. It is for inbound traffic on a Cisco 3400 router.

```
policy-map qos-in-$Interface_Name
class class-default
#if($PIR_in_mbps==0)
     police cir $CIR_in_mbps m
#elseif($PIR_in_mbps!=0)
     police cir $CIR_in_mbps m pir $PIR_in_mbps m
#end

!
interface $Interface_Name
service-policy input qos-in-$Interface_Name
```

Template Data File

Here is the template date file to be attached to policy:

```
CIR_in_mbps:   $CIR_in_mbps
PIR_in_mbps:   $PIR_in_mbps
Interface_Name: $UNI_INTERFACE_NAME
```

Additional Attribute Definition File

Here is the additional information definition file:

```
<additionalInformation>
<group name="QoS">
        <attribute>
                <name>$CIR_in_mbps</name>
                <value></value>
                <displayName>Committed Bandwidth</displayName>
                <type>
                        <integer>
                                <lower>1</lower>
                                <upper>32000</upper>
                                <rangeUnits>Mbps</rangeUnits>
                        </integer>
                </type>
                <description>CIR value in Mbps</description>
                <required>true</required>
        </attribute>
        <attribute>
                <name>$PIR_in_mbps</name>
```

```
                <value></value>
                <displayName>Peak Bandwidth</displayName>
                <type>
                        <integer>
                                <lower>1</lower>
                                <upper>32000</upper>
                                <rangeUnits>Mbps</rangeUnits>
                        </integer>
                </type>
                <description>PIR value in Mbps</description>
                <required>false</required>
        </attribute>
</group>
</additionalInformation>
```

Additional Attributes Displayed in the Service Request Workflow

Based on this example, two new attributes are displayed in the service request workflow:

- Committed Bandwidth
- Peak Bandwidth

Committed Bandwidth is a required field, and Peak Bandwidth is an optional field.

User Input and Sample Configlets

The following examples show user input for the new attributes and the resulting configlets that are generated.

Example 1

User input:

- Committed Bandwidth: 25

Configlet generated:

```
policy-map qos-in-<uni interface>
class class-default
    police cir 25m
!
interface <uni interface>
service-policy input qos-in-<uni interface>
```

Example 2

User input:

- Committed Bandwidth: 25
- Peak Bandwidth: 50

Configlet generated:

```
policy-map qos-in-<uni interface>
class class-default
```

```
    police cir 25 m pir 50 m
!
interface <uni interface>
service-policy input qos-in-<uni interface>
```