Using Templates

IPsec, firewall, NAT: **These features are not supported in this release**.

Cisco IP Solution Center (ISC) uses templates to generate device commands that are not supported by ISC, and to download them to a Cisco device. For example, ISC does not configure importing of root certificates. A template enables you to add this configuration to a device. A template configuration file can be either a partial or complete configuration file. The template configuration file is merged with (either appended or prepended to) the ISC configlet. The combined configlet is downloaded to the device as part of a service request or as a transient template.

Templates are defined in service definitions and can be deployed:

- Using a service order.
- Attached to a service request for another service (see the "Templates in a Service Request" section on page 4-9).

You can use the API to generate template definitions, template data, and device configlets based on the templates.

This chapter contains the following sections:

Refer to the *Cisco IP Solution Center Integrated VPN Management Suite Infrastructure Reference, 4.0* for information on the GUI Template Manager.

Template Overview

Templates consist of template definitions and template data. The template definition contains the logic and variables to be populated with template data. The template data is the configuration information to be downloaded to a device. When ISC merges the template definition's variables with the data in the template data file, a template configuration file is created. The template configuration file is downloaded to the device.

Templates can be deployed independently of other ISC functions or they can be attached to a service request.

The API supports the following types of template operations:

- Templates created from a template definition and a data file.
- Buffer templates—Template data is pulled from a data buffer instead of a data file and inserted directly into a service request (only for MPLS service requests).
- Templates integrated as part of a service request—The service request specifies the device to receive the configuration (the template definition and template data method).
- Transient templates—Transient template data is used only for the download and then discarded. It is not available for subsequent viewing (only for direct template download service requests).

Template definition files and template data files are stored in XML format. The template definition file, its data files, and all resulting template configuration files are mapped to a single directory. One template definition can contain many data files, but a template data file can be attached to only one template definition.

Tip When you generate a template configuration file using a particular template data file, the configuration filename correlates to the data filename.

To view the interaction between the template and the device, use the task logs. See the "Viewing Task Logs" section on page 5-23 for more information.

Template Definition

The template definition defines the variables that are populated with template data. It defines the actions that need to be taken for any device to which the template is attached.

The template definition specifies what data is necessary to create the template configuration file, and includes how the variable names and the data are associated.

Note The template definition in the API corresponds to the template in the GUI.

Template Data

The template data consists of name/value pairs for each variable defined in the template definition. Each template data file can be associated with only one template definition.

Template data can be created using the GUI or the API. The data can exist in a template data file, be merged with a template definition from a data buffer, or be entered as transient data directly into a service request.

Creating a template data file is a separate operation. However, if you use transient data or data buffers, this allows you to enter template data at the same time you are creating the service definition or service request.

The data file contains data for all variables in the template definition.

Note To view the configuration created using a template, without downloading the template to the device, use the ViewTemplateConfig XML request. Specify the template definition and template data, and the configuration is returned in the XML response.

Template Operations

Template definitions and template data files are specified in a service definition. The device to receive the template configuration and transient data and data buffers (if applicable) are defined in the service request as part of a service order.

The API supports these template subtypes:

- **TemplateDefinition**—The template itself, which contains the variables and logic to be populated with template data.
- **TemplateData**—The data to be merged with a template definition.
- **TemplateConfig**—The template configlet that is the result of the template definition being merged with the template data.
- **TemplateDownload**—Used to download a template configlet to a device using template data from a data file.
- **TemplateTransient**—Used to download a template configlet to a device using template data that is added directly into the XML request.

The following template operations can be executed using the API:

- For service definition subtypes:
  - **TemplateDefinition**—Create, Delete, Modify, or View
  - **TemplateData**—Create, Delete, Modify, or View
- For service request subtypes:
  - **TemplateConfig**—Create, Modify, or View
  - **TemplateDownload**—Create, Delete, Modify, or View
  - **TemplateDataTransient**—Create or View

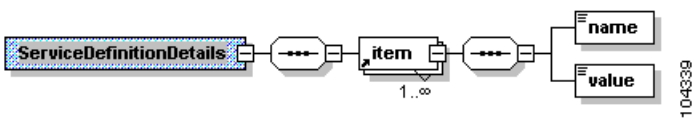
Template Service Definitions

Using service definitions to define templates enables the XML requests to be processed the same as other service policies (QoS, MPLS, and L2VPN) and allows them to be specified in service orders.

A template service definition consists of a template definition and the corresponding template data items. The template definition specifies the variable names and logic in the **BodyText** property.

Figure 4-1 shows the schema diagram for a template service definition. The *item* refers to the template definition, and the *name/value* pairs refer to the template data.

Figure 4-1 Schema Diagram for Template Service Definitions



Template Service Orders

A template is implemented using a service order. During a service request deployment, the template definition and data file are merged, and the resulting configuration is appended or prepended to the ISC-generated configlet. The combined configuration is downloaded to the device specified in the service request.

If the template is:

- Prepended—The template commands take place before the service request commands.
- Appended—The template commands take place after the service request commands.

Service orders can specify template downloads, transient data downloads, and templates specified within a service request.

To view a template service order:

- For templates that specify a data file, only the data file name is listed in the service request. Viewing the data file is a separate operation.
- For templates that specify a data buffer, the data is displayed within the service request. Template data buffers can only be viewed by viewing the service request. Use the **enumerateInstances** operation and enter the **LocatorId** of the service request that contains the template data buffers.

Provisioning Example

This section describes the required steps for using templates independent of service requests. See the "Templates in a Service Request" section on page 4-9 for information on deploying templates with service requests.

Prerequisites

ISC provides pre-populated examples to help you create a template.

- If you are using Sybase as a back-end database, you are provided with pre-populated template examples. These examples can be found on the left pane of the main Template Manager window.
- If you are using Oracle as a back-end database, you are NOT provided with pre-populated template examples. You must either create a template definition from scratch or import a template. Alternately, you can run the script **populateTemplates.sh** located in the **<install-dir>/bin** directory.

Process Summary

In this template provisioning example, the following steps are listed:

- Create a template definition file.
- Create the template data file.
- View the template configuration.
- Download the template configuration to a device.
- Delete the template data file and template definition.

Provisioning Process

This section provides an example provisioning process using XML examples. The inventory of XML examples for the ISC API can be found at the following location:

http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/isc/4_0/api/apiref/examples/index.htm

Step 1 Create a template definition.

Table 4-1 Create Template Definition

Operation	className	Required Parameters
createInstance	ServiceDefinition	• Name • Type=TemplateDefn • ServiceDefinitionDetails
	ServiceDefinitionDetails	Can contain one or more of the following classes, with associated variable name/value pairs as child objects: • TemplateInteger • TemplateString

XML Example:

- CreateTemplateDefnSimple.xml

Step 2 Create a template data file.

Table 4-2 Create Template Data

Operation	className	Required Parameters
createInstance	ServiceDefinition	• Name • Type=TemplateData • ServiceDefinitionDetails
	ServiceDefinitionDetails	<template data> (The name/value pairs.)

The following XML examples show how to populate a template containing one-dimensional variables or two-dimensional variables. The incoming template data must match the format of the template definition. The API validates the incoming data against the variable definition. An error is returned if they do not match.

XML Examples:

- CreateTemplateData1Dim.xml
- CreateTemplateData2Dim.xml

Step 3 View the template data file.

To view the data file, provide the full path name and filename. This is the same as the folder path name and filename in the GUI.

Table 4-3 View Template Data

Operation	className	Required Parameters
enumerateInstances	ServiceDefinition	• Name=<pathname/filename to template data file> • Type=TemplateData

XML Example:

ViewTemplateData.xml

Step 4 View the configlet generated for the template.

Table 4-4 View Configlet

Operation	className	Required Parameters
enumerateInstances	Task	• Type=TemplateConfig • TemplateDefn=<pathname to template definition> • TemplateData=<template data filename>

XML Example:

ViewTemplateConfig.xml

The following example shows a response to a ViewTemplateConfig XML request.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://www.cisco.com/cim-cx/2.0" xmlns:ns1="urn:CIM">
  <soapenv:Header>
    <ns0:message id="87855" sessiontoken="F1856684E9A183F5E542890772B3D040"
timestamp="2003-09-25T21:38:07.645Z" />
  </soapenv:Header>
  <soapenv:Body>
    <ns1:enumerateInstancesResponse>
      <returns xsi:type="ns1:CIMPropertyList" soapenc:arrayType="ns1:CIMProperty[]">
        <item xsi:type="ns1:CIMProperty">
          <name xsi:type="xsd:string">Configlet</name>
          <value xsi:type="xsd:string">
ip vrf $vrfName
 maximum routes 2 75
  export map To_NM_VPN
 route-target import 2:103000
 route-target import 2:103500
 route-target import 2:103020
 route-target import 2:103520
 route-target import 0
```

```
 route-target import 1
 exit
router bgp 13979
 address-family ipv4 vrf vrf1
 default-information originate
 maximum-paths eibgp 6
 bgp suppress-inactive
   neighbor 10.10.10.1 route-map set_ce_local_pref in
 neighbor 10.10.10.1 maximum-prefix 21 50
 neighbor 10.10.10.1 capability orf prefix-list receive
 neighbor 10.10.10.1 advertisement-interval 60
    exit
</value>
        </item>
      </returns>
    </ns1:enumerateInstancesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Step 5 Download the template configuration to a device.

Table 4-5 Download Template Configuration

Operation	className	Required Parameters
performBatchOperation		
createInstance	ServiceOrder	• ServiceName • NumberOfRequests • ServiceRequest
	ServiceRequest	• RequestName • Type=TemplateDownload • Template definition information: – ServiceDefinition=<pathname to template definition> – ServiceDefinitionType= TemplateDefn • Template data information: – ServiceDefinition=<pathname /filename to template data file> – ServiceDefinitionType= TemplateData • ServiceRequestDetails
	ServiceRequestDetails	LogicalDevice=<name of device to receive the template configuration>

XML Example:

CreateTemplateServiceOrderDownload.xml

Step 6 Delete the template data file and the template definition file. This step is optional.

Table 4-6 Delete Template Files

Operation	className	Required Parameters
deleteInstance	ServiceDefinition	To delete the template data file: • Name=<pathname to template definition file> • Type=TemplateData To delete the template definition file: • TemplatePathname=<pathname/file name to template data file> • Type=TemplateDefn

XML Examples:

- DeleteTemplateData.xml
- DeleteTemplateDefn.xml

Transient Templates

For transient templates, the template data is not specified through a previously defined data file. The template data is entered directly into the XML request. Transient data is used only for the instance of the service order and is then discarded. The transient data is not available for subsequent service orders, and you cannot view transient data when you view the service order.

- To view the generated configlet, refer to "View the configlet generated for the template." on page 6.
- To download transient template data to a device, refer to "Download the template configuration to a device." on page 7.

For transient templates, leave out the following two properties:

- **ServiceDefinition=<pathname/filename to template data file>**
- **ServiceDefinitionType=TemplateData**

Instead, specify **SubType=TemplateDataTransient** in the **ServiceDefinition**, and enter the template data (name/value pairs) in the **ServiceDefinitionDetails**. See the following example:

```
<objectPath xsi:type="ns1:CIMObjectPath">
      <className xsi:type="xsd:string">ServiceRequest</className>
      <properties xsi:type="ns1:CIMPropertyList"
                  soapenc:arrayType="ns1:CIMProperty[]">
       <item xsi:type="ns1:CIMProperty">
        <name xsi:type="xsd:string">RequestName</name>
        <value xsi:type="xsd:string">MYSR-1</value>
       </item>
       <item xsi:type="ns1:CIMProperty">
        <name xsi:type="xsd:string">Type</name>
        <value xsi:type="xsd:string">TemplateDownload</value>
       </item>
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">ServiceDefinition</name>
```

```
  <value xsi:type="xsd:string">/User/UsernameTemplate</value>
  <qualifier xsi:type="ns1:CIMQualifier">
    <name xsi:type="xsd:string">ServiceDefintionType</name>
    <value xsi:type="xsd:string">TemplateDefn</value>
  </qualifier>
</item>
<objectPath xsi:type="ns1:CIMObjectPath">
 <className xsi:type="xsd:string">ServiceDefinition</className>
 <properties xsi:type="ns1:CIMPropertyList"
           soapenc:arrayType="ns1:CIMProperty[]">
 <item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">SubType</name>
    <value xsi:type="xsd:string">TemplateDataTransient</value>
 </item>
 </properties>
 <objectPath xsi:type="ns1:CIMObjectPath">
  <className xsi:type="xsd:string">ServiceDefinitionDetails</className>
  <properties xsi:type="ns1:CIMPropertyList"
           soapenc:arrayType="ns1:CIMProperty[]">
   <item xsi:type="ns1:CIMProperty">
    <name xsi:type="xsd:string">username</name>
    <value xsi:type="xsd:string">user1</value>
   </item>
 </properties>
 </objectPath>
```

XML Example:

CreateTemplateServiceOrderDownloadTransient.xml

Templates in a Service Request

You can add templates to a service request. When the service order is deployed, the template configuration is downloaded to the device, along with the configuration from the service request.

Note To remove templates from a service request, see "Removing Template Configurations" section on page 4-13.

The template information in a service request template contains the **LinkTemplate** object, which defines the location of the template definition and data files, the device to receive the configuration download, and whether to prepend or append the template configuration.

Link Template

When you include templates in a service request, the template information is defined using the **LinkTemplate** object. The **LinkTemplate** contains the path to the template definition and the location of the template data.

Note IPsec service requests use **IPSecTemplate** for site-to-site services, and **IPSecRATemplate** for remote access services.

For each service type, the **LinkTemplate** is defined in these link objects in the service request:

- MPLS—**MplsVpnLink**
- L2VPN—**ACAttr** (EndtoEndWire>AttachmentCircuit>ACAttr)
- VPLS—**VPLSLink**
- Firewall—**FirewallLink**
- NAT—**NATLink**
- IPsec—**IPSecLink** (For IPsec service requests, use **IPSecTemplate** or **IPSecRATemplate** in place of LinkTemplate.)

Note See the appropriate chapter on service provisioning for more information on using LinkTemplate in service requests.

Data Buffer Object

If the template data is pulled from a template data file, the **LinkTemplate** object contains the path to the data file. If the template data is pulled from a data buffer (MPLS only), the **LinkTemplate** object contains the **DataBuffer** object. The **DataBuffer** can contain values for any variable defined in the template definition.

In the following example, the values for the variables **Source-IP**, **Dest-IP**, and **protocol**, are defined in the **DataBuffer** object.

```
<objectPath xsi:type="ns1:CIMObjectPath">
            <className xsi:type="xsd:string">DataBuffer</className>
            <properties xsi:type="ns1:CIMPropertyList"
                        soapenc:arrayType="ns1:CIMProperty[]">
              <item xsi:type="ns1:CIMProperty">
                  <name xsi:type="xsd:string">Source-IP</name>
                  <value xsi:type="xsd:string">132.235.123.0</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                  <name xsi:type="xsd:string">Dest-IP</name>
                  <value xsi:type="xsd:string">54.103.63.0</value>
              </item>
             <item xsi:type="ns1:CIMProperty">
                  <name xsi:type="xsd:string">protocol</name>
                  <value xsi:type="xsd:string">udp</value>
              </item>
```

You can also use the **DataBuffer** to specify values for variables defined elsewhere in the service request. Instead of entering the variable and value in the service request and then repeating them again in the **LinkTemplate**, you can simply call the value using the **DataBuffer**.

In the following partial example for an MPLS service request, in the **LinkAttrs** class, values are listed for **PE_VCI**, **PE_BGP_AS_ID**, and **Max_route_threshold**.

```
<objectPath xsi:type="ns1:CIMObjectPath">
               <className xsi:type="xsd:string">LinkAttrs</className>
                <properties xsi:type="ns1:CIMPropertyList"
                       soapenc:arrayType="ns1:CIMProperty[]">
                <item xsi:type="ns1:CIMProperty">
                     <name xsi:type="xsd:string">CE_Intf_Name</name>
                     <value xsi:type="xsd:string">ATM1.22</value>
                </item>
                <item xsi:type="ns1:CIMProperty">
```

```
                    <name xsi:type="xsd:string">PE_Intf_Name</name>
                    <value xsi:type="xsd:string">Switch1.234</value>
                 </item>
                <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">PE_VCI</name>
                    <value xsi:type="xsd:string">234</value>
                 </item>
                <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">PE_BGP_AS_ID</name>
                    <value xsi:type="xsd:string">13979</value>
                 </item>
                <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">Max_route_threshold</name>
                   <value xsi:type="xsd:string">25</value>
               </item>
             </properties>
           </objectPath>
```

The next section of this example shows these same values being called again in the **DataBuffer**.

```
<objectPath xsi:type="ns1:CIMObjectPath">
            <className xsi:type="xsd:string">DataBuffer</className>
            <properties xsi:type="ns1:CIMPropertyList"
                        soapenc:arrayType="ns1:CIMProperty[]">
              <item xsi:type="ns1:CIMProperty">
                  <name xsi:type="xsd:string">PE_VCI</name>
                  <value xsi:type="xsd:string">$PE_VCI</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                  <name xsi:type="xsd:string">PE_BGP_AS_ID</name>
                  <value xsi:type="xsd:string">$PE_BGP_AS_ID</value>
              </item>
              <item xsi:type="ns1:CIMProperty">
                  <name xsi:type="xsd:string">Max_route_threshold</name>
                  <value xsi:type="xsd:string">$Max_route_threshold</value>
              </item>
```

Note See the Repository Variable Chooser in the GUI Template Manager for a list of variables, by service blade, that can be recalled by the **DataBuffer**. (From the Service Design tab, click the Templates link. Choose the template Data file and click **Vars** on the Data File Editor window.)

In Table 4-7, the required parameters listed for **ServiceRequestDetails** are only for the template portion of the service order. For more information on service requests, see the appropriate chapters on service provisioning in this guide.

Table 4-7 Templates in a Service Request

Operation	className	Required Parameters
createInstance	ServiceOrder	• ServiceName • NumberOfRequests • ServiceRequest
	ServiceRequest	• RequestName • Type=<choose the appropriate service type> • ServiceRequestDetails

Table 4-7 Templates in a Service Request (continued)

Operation	className	Required Parameters
	ServiceRequestDetails	• MplsVpnLink (or **ACAttr**, **IPSecLink**, **FirewallLink**, **NATLink**)
	MplsVpnLink (or the link object for your service)	• LinkTemplate **Note** For IPsec, use **IPSecTemplate** for site-to-site services, and **IPSecRATemplate**, for remote access services.
	LinkTemplate (or **IPSecTemplate**, **IPSecRATemplate**)	• DatafilePath=<the pathname to the template definition folder> • LogicalDevice • The template data information, either from a data file or a data buffer. – DatafileName=<the pathname/filename to the template data file> – DataBuffer=<template data> (The name/value pairs.) • TemplateActive=true • TemplateAction= – APPEND – PREPEND

Note The attributes **PE_Template**, **PE_Intf_Template**, **CE_Template**, and **CE_Intf_Template** allow NBI access to variables designed to hold template blobs (template blobs were used during MPLS provisioning in legacy versions of ISC).

XML Examples:

- CreateMPLSTemplateServiceOrder.xml
- CreateL2VPNTemplateServiceOrder.xml
- CreateFWServiceOrderwTemplate.xml
- CreateIPSecServiceOrder.xml
- CreateIPSecRAServiceOrder.xml
- CreateNATServiceOrderwTemplate.xml

Removing Template Configurations

When you modify a service request that has templates, or before you can decommission a service request that has templates, you must first remove the template information from the service request. This is accomplished using negate templates to remove the template configuration from the device.

Modifying a Service Request with Templates

To modify a template in an existing service request, the following tasks must occur in the order listed:

1. Turn off templates. This action changes the **TemplateActive** attribute for the template from **true** to **false**.
   - For MPLS service requests, the **modifyInstance** subaction automatically toggles the **TemplateActive** attribute to **false**.
   - For all other service requests, the **TemplateActive** attribute must be specifically set to **false** in the **modifyInstance** subaction.
2. Add negate templates. This action removes the template information from the device. Use the **createInstance** subaction to add a negate template.
3. Add new templates. This action adds a new template to the service request. Use the **createInstance** action to add templates.

Note You should wait until the task has completed (you receive a task completed message) before you run the next task.

Turning off Templates (for MPLS Service Requests)

When you create the MPLS service request with a template, ISC sets the attribute **TemplateActive=true**. To turn off the template, the attribute needs to be changed to **TemplateActive=false**.

For templates in an MPLS service request, this is accomplished using a **modifyInstance** subaction. When you execute a **modifyInstance** subaction on a template, ISC automatically changes the status of the attribute to **TemplateActive=false**.

Note This automatic change in the template attribute only occurs when you use a **modifyInstance** on a template in an MPLS service request. For other service type, see the "Turning off Templates (for All Other Service Types)" section on page 4-14.

Include the device that received the template configuration and the template name (**DataFilePath**) from the service request where the template was implemented. See the following example:

```
<objectPath subAction="modifyInstance" xsi:type="ns1:CIMObjectPath">
               <className xsi:type="xsd:string">LinkTemplate</className>
               <keyProperties xsi:type="ns1:CIMKeyPropertyList"
                      soapenc:arrayType="ns1:CIMKeyProperty[]">
                 <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">LogicalDevice</name>
                   <value xsi:type="xsd:string">PE-POP1</value>
                 </item>
                 <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">DatafilePath</name>
```

```
                    <value xsi:type="xsd:string">/Examples/templ1-enable</value>
                  </item>
```

In this XML example, the template name /Examples/temp11-enable, for device PE-POP1, is turned off (**TemplateActive=false**) by the **modifyInstance** subaction.

Turning off Templates (for All Other Service Types)

Unlike with MPLS, this attribute change does not happen automatically for all other service types. You must use a **modfyInstance** subaction and include the attribute **TemplateActive=false** in the XML request. See the following example:

```
<objectPath subAction="modifyInstance" xsi:type="ns1:CIMObjectPath">
               <className xsi:type="xsd:string">LinkTemplate</className>
               <keyProperties xsi:type="ns1:CIMKeyPropertyList"
                      soapenc:arrayType="ns1:CIMKeyProperty[]">
                 <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">LogicalDevice</name>
                   <value xsi:type="xsd:string">PE-POP1</value>
                 </item>
               <item xsi:type="ns1:CIMProperty">
                    <name xsi:type="xsd:string">TemplateActive</name>
                    <value xsi:type="xsd:string">false</value>
                </item>
               <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">DatafilePath</name>
                   <value xsi:type="xsd:string">/Examples/templ1-enable</value>
                 </item>
```

Adding Negate Templates

After the template is turned off (changed to **TemplateActive=false**), you add a negate template using a **createInstance** subaction to remove the template information from the device. The **DataFilePath** of the negate template must be different from the original template.

See the following example:

```
<objectPath subAction="createInstance" xsi:type="ns1:CIMObjectPath">
              <className xsi:type="xsd:string">LinkTemplate</className>
               <keyProperties xsi:type="ns1:CIMKeyPropertyList"
                     soapenc:arrayType="ns1:CIMKeyProperty[]">
               </keyProperties>
                <properties xsi:type="ns1:CIMPropertyList"
                      soapenc:arrayType="ns1:CIMProperty[]">
               <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">LogicalDevice</name>
                   <value xsi:type="xsd:string">PE-POP1</value>
               </item>
               <item xsi:type="ns1:CIMProperty">
                   <name xsi:type="xsd:string">DatafilePath</name>
                   <value xsi:type="xsd:string">/Examples/tempnegate</value>
               </item>
```

Adding New Templates

When the original template information is disabled and removed from the device, you can add new template information using:

- A **createInstance** to create the service order to modify the service request.

- A **modifyInstance** to modify the service request and service request details.
- A **createInstance** subaction to add the new template.

Note You are not required to create a new service order to add new templates. In one modify service request, you can turn off templates, add negate templates, and add new templates. However, you must keep the correct order of operations (turn off, add negate, then add new).

See the following example:

```
<ns1:performBatchOperation>
    <actions xsi:type="ns1:CIMActionList"
             soapenc:arrayType="ns1:CIMAction[]">
     <action>
      <actionName xsi:type="xsd:string">createInstance</actionName>
      <objectPath xsi:type="ns1:CIMObjectPath">
       <className xsi:type="xsd:string">ServiceOrder</className>
       <properties xsi:type="ns1:CIMPropertyList"
                   soapenc:arrayType="ns1:CIMProperty[]">
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">ServiceName</name>
         <value xsi:type="xsd:string">Acme-Template1</value>
        </item>
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">CarrierId</name>
         <value xsi:type="xsd:string">22</value>
        </item>
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">DesiredDueDate</name>
         <value xsi:type="xsd:dateTime">2002-12-13T14:55:38.885Z</value>
        </item>
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">Organization</name>
         <value xsi:type="xsd:dateTime">NbiCustomer</value>
        </item>
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">NumberOfRequests</name>
         <value xsi:type="xsd:string">1</value>
        </item>
       </properties>
      </objectPath>
     </action>
     <action>
      <actionName xsi:type="xsd:string">modifyInstance</actionName>
      <objectPath subAction="modifyInstance" xsi:type="ns1:CIMObjectPath">
      <className xsi:type="xsd:string">ServiceRequest</className>
      <properties xsi:type="ns1:CIMPropertyList"
                  soapenc:arrayType="ns1:CIMProperty[]">
       <item xsi:type="ns1:CIMProperty">
        <name xsi:type="xsd:string">RequestName</name>
        <value xsi:type="xsd:string">Template1</value>
       </item>
        <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">Type</name>
         <value xsi:type="xsd:string">Mpls</value>
        </item>
       </properties>
       <objectPath subAction="modifyInstance" xsi:type="ns1:CIMObjectPath">
           <className xsi:type="xsd:string">ServiceRequestDetails</className>
            <keyProperties xsi:type="ns1:CIMKeyPropertyList"
                      soapenc:arrayType="ns1:CIMKeyProperty[]">
                  <item xsi:type="ns1:CIMProperty">
```

```
          <name xsi:type="xsd:string">LocatorId</name>
          <value xsi:type="xsd:string">36</value>
        </item>
  </keyProperties>
  <objectPath subAction="modifyInstance" xsi:type="ns1:CIMObjectPath">
   <className xsi:type="xsd:string">MplsVpnLink</className>
    <keyProperties xsi:type="ns1:CIMKeyPropertyList"
          soapenc:arrayType="ns1:CIMKeyProperty[]">
       <item xsi:type="ns1:CIMProperty">
         <name xsi:type="xsd:string">LocatorId</name>
         <value xsi:type="xsd:string">33</value>
       </item>
    </keyProperties>
    <properties xsi:type="ns1:CIMPropertyList"
          soapenc:arrayType="ns1:CIMProperty[]">
    </properties>
    <objectPath subAction="modifyInstance" xsi:type="ns1:CIMObjectPath">
   <className xsi:type="xsd:string">LinkAttrs</className>
    <keyProperties xsi:type="ns1:CIMKeyPropertyList"
          soapenc:arrayType="ns1:CIMKeyProperty[]">
   </keyProperties>
   <properties xsi:type="ns1:CIMPropertyList"
          soapenc:arrayType="ns1:CIMProperty[]">
   </properties>
  </objectPath>
 <objectPath subAction="createInstance" xsi:type="ns1:CIMObjectPath">
   <className xsi:type="xsd:string">LinkTemplate</className>
   <keyProperties xsi:type="ns1:CIMKeyPropertyList"
          soapenc:arrayType="ns1:CIMKeyProperty[]">
     <item xsi:type="ns1:CIMProperty">
       <name xsi:type="xsd:string">LogicalDevice</name>
       <value xsi:type="xsd:string">PE-POP1</value>
     </item>
     <item xsi:type="ns1:CIMProperty">
       <name xsi:type="xsd:string">DatafilePath</name>
       <value xsi:type="xsd:string">/Examples/templ4-enable</value>
     </item>
   </keyProperties>
  <properties/>
  <!-- objectPath xsi:type="ns1:CIMObjectPath">
<className xsi:type="xsd:string">DataBuffer</className>
<properties xsi:type="ns1:CIMPropertyList"
          soapenc:arrayType="ns1:CIMProperty[]">
  <item xsi:type="ns1:CIMProperty">
      <name xsi:type="xsd:string">DLCI</name>
      <value xsi:type="xsd:string">20</value>
  </item>
```

Decommissioning a Service Request

When you decommission a service request with templates, you must first remove the template information from the service request.

The following processes must occur, in the order listed, to decommission a service request:

- Modify the service request to turn off (**TemplateActive=false**) the templates. Refer to the appropriate section for information:
  - "Turning off Templates (for MPLS Service Requests)" section on page 4-13
  - "Turning off Templates (for All Other Service Types)" section on page 4-14

- Create a service order to decommission the service request. Refer to the "Service Decommission" section on page 3-10 for more information.