Configuration Templates Management

This chapter details the templates that Cisco Broadband Access Center (BAC) supports for device configuration and device management.

This chapter includes the following sections:

- Overview, page 5-1
- Features of Cisco BAC Templates, page 5-4
- Authoring Configuration Templates, page 5-14
- Using the Configuration Utility, page 5-22

Overview

Cisco BAC provides an extensible template-based mechanism to assign configurations for devices that are compliant with the CPE WAN Management Protocol (CWMP). This template-processing mechanism simplifies configuration management, enabling customized configurations for millions of customer premises equipment (CPE) by using a small number of templates.

The result of processing a template is an instruction, which is forwarded to the DPEs in a device's provisioning group. This instruction contains all the necessary information for the DPE to configure the device.

Before attempting to create your own template file, familiarize yourself with information on:

- Parameter dictionaries (see Parameter Dictionaries, page 7-1)
- Firmware templates (see Firmware Management, page 6-1)

By using Cisco BAC templates, you can create a template file in an easily readable format, and edit it quickly and simply. You must add template files to the RDU by using the administrator user interface or the API, before any Class of Service can reference it.

Cisco BAC templates are XML documents written according to a published schema. When instructions are generated for a given device at the RDU for distribution to DPEs, a common template processor retrieves a template by using the device’s Class of Service object.

It then evaluates the template values, such as Includes and Conditionals, and substitutes template variables with their values. The resulting output is an XML object which represents a processed template.

The syntax and content of the processed configuration template is validated when it is added to the Cisco BAC system, ensuring that the XML template is well-formed. This validation also ensures that parameter names and values are consistent with definitions in the parameter dictionary, an XML file that defines the supported objects and parameters. (See Parameter Dictionaries, page 7-1.)

Note This type of validation occurs only when a template is added to the system or when the content of an existing template is replaced

Cisco BAC uses the XML schemas that are defined in various files to generate instructions for device configurations. Table 5-1 lists these files and their locations.

Table 5-1 Files Used in Configuration Template Processing

File	Purpose	Options Available in Cisco BAC
Configuration Template Samples	Defines device configuration	Sample templates
	Sample templates are located at: <BPR_HOME>/rdu/samples/cwmp	
Configuration Template Schema	Validates configuration template syntax	Default template schemas
	Default template schemas are located at: • CWMP configuration schema <BPR_HOME>/rdu/templates/cwmp/schema/CwmpTemplateConstructs.xsd • Common template schema <BPR_HOME>/rdu/templates/cwmp/schema/CommonTemplateConstructs.xsd	
Parameter Dictionary	Validates configuration template content	Default dictionaries Custom dictionaries
	Default dictionaries are located at: • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr069-cwmp-dictionary.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr098-cwmp-dictionary.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr104-cwmp-dictionary.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr106-cwmp-dictionary.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr196-cwmp-dictionary-v1.1.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr196-cwmp-dictionary-v2.0.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr196-cwmp-dictionary-IGD-v1.1.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/tr196-cwmp-dictionary-IGD-v2.0.xml • <BPR_HOME>/rdu/templates/cwmp/dictionary/basic-cwmp-dictionary.xml	
Parameter Dictionary Schema	Validates parameter dictionary syntax	Default dictionary
	Parameter dictionary schema is located at: Schema for TR-069, TR-098, TR-104, TR-106, TR-181, and TR-196 dictionaries <BPR_HOME>/rdu/templates/cwmp/schema/TemplateDictionarySchema.xsd	

Cisco BAC supports multiple instance object which is available in the configuration template. This facilitates discovering and updating parameters for the selective object instances without specifying the object instance number. For more information on multi-instance object support see, Multi-Instance Object Support, page 4-5.

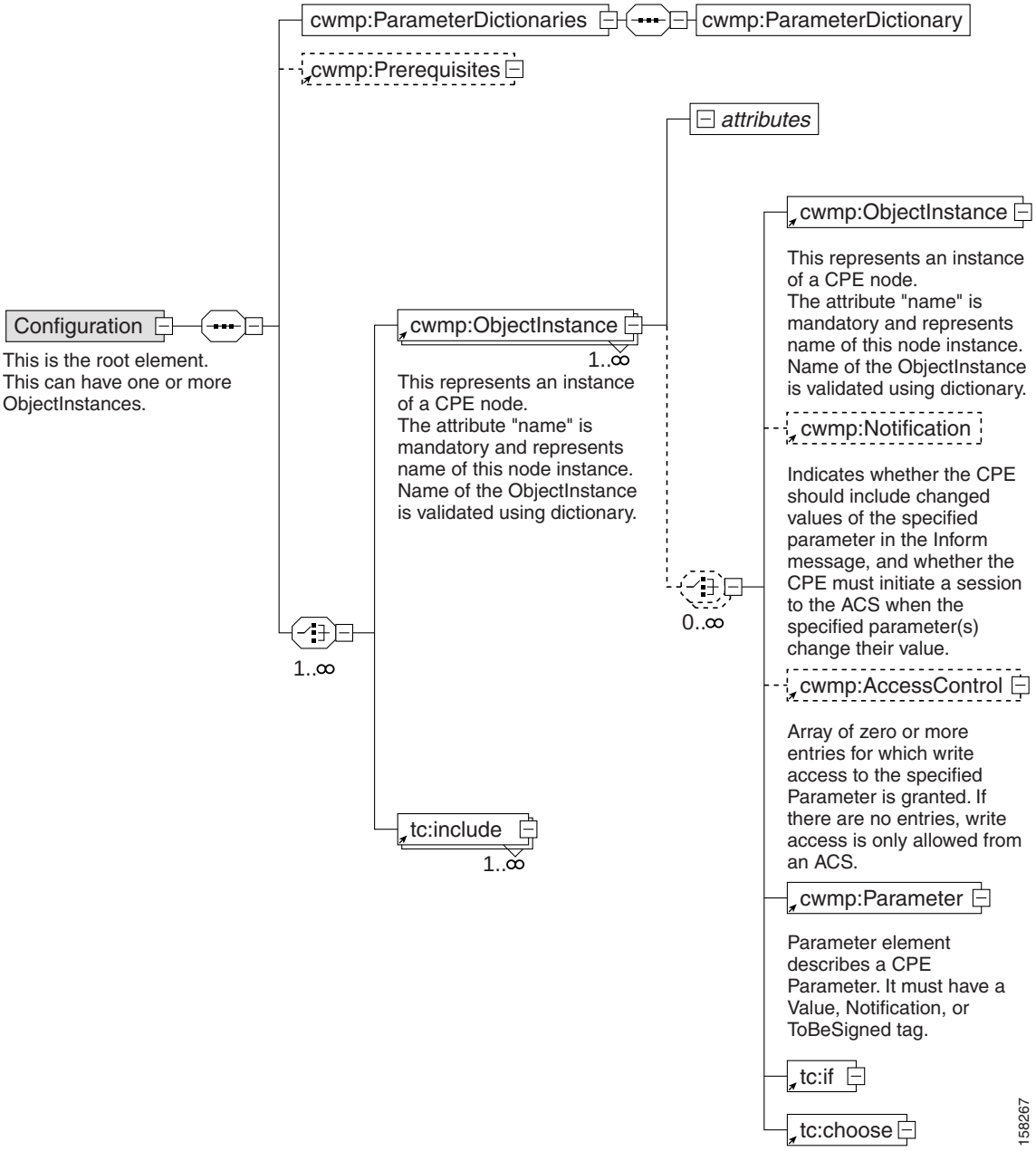
You can manage template files, configuration and firmware rules, by using the administrator user interface.

Features of Cisco BAC Templates

A configuration template is a collection of Objects and Parameters. In a TR-069 device, *InternetGatewayDevice* is the root object, while in TR-106 device, *Device* is the root object. In a TR-196 device, *InternetGatewayDevice* or *Device* is the root object.

Figure 5-1 illustrates the schema of a TR-069 device Configuration. Each element featured in the schema is described in subsequent sections.

Figure 5-1 Configuration Schema



A configuration template comprises the following components:

- **ObjectInstance**—Represents an instance of a TR-069 CPE node. You must specify the object's 'name.'

  An object may include other objects and parameters, both of which, in turn, may include other elements in line with the TR-069 specification.

  A CPE object may contain:

  - Object—Describes an instance of a CPE node.
  - Parameters—Describes a CPE parameter and must have a value, and Notification, Access Control, or both. (See Parameters, page 5-6.)
  - Notification—Enables notification of changes in the value on all child parameters of this parameter at all levels. (See Notification, page 5-8.)
  - Access Control—Controls whether or not entities other than the autoconfiguration server (user, SNMP, UPnP, and so on) can change values of any configuration parameters under this object. (See Access Control, page 5-9.)
- **Parameter Dictionary**—Contains definitions used to validate the objects and parameters in configuration templates. Cisco BAC supports only one dictionary per template. (See Parameter Dictionaries, page 7-1.)
- **Prerequisites**—Optionally, indicates the conditions that must be met before a device can be configured. These conditions may include:

  - MaintenanceWindow—Enables you to specify the time that a configuration is to be applied to a device.
  - Expressions—Enable you to specify any parameters that need to match on the device as a prerequisite to applying the configuration; for example, a particular software or hardware version of a device.

  See Figure 5-3 for a graphical representation of the prerequisite schema. For detailed information on using the prerequisite option, see Prerequisites, page 5-9.

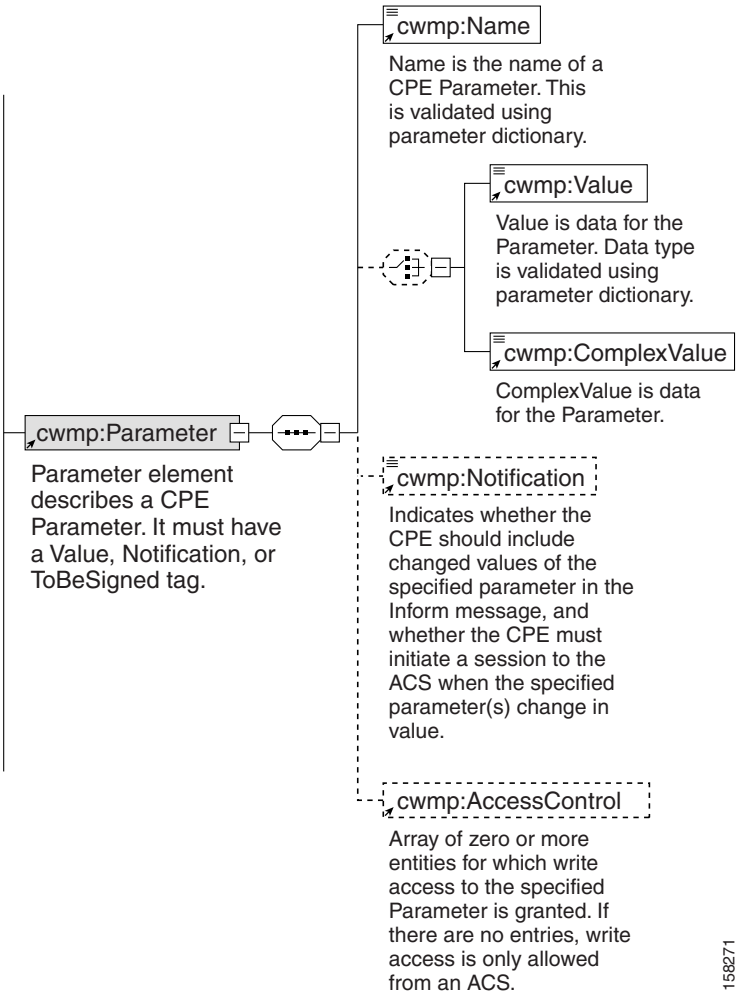
Adding a configuration template to Cisco BAC would fail if there are errors in the template. To avoid configuration errors, ensure that all:

- Object names and parameter names exist in the parameter dictionary.
- Parameter values are of the type specified in the parameter dictionary.
- Parameters that are not writable do not have a value. However, they may have the Notification, Access Control attributes, or both set.
- Substitutable variables are defined in the system through Cisco BAC custom property or device property.

Parameters

This section describes the schema associated with parameter objects in the configuration templates (see Figure 5-2).

Figure 5-2 Parameter Schema



The CPE parameter contains a name-value pair:

The name identifies the parameter, and has a hierarchical structure similar to files in a directory, with each level separated by a dot (.). Each level corresponds to an object instance: Some objects have a single instance or singletons, other objects have multiple instances. The parameter lists for both object instances are described in subsequent sections.

The value of a parameter may be one of several data types defined in the TR-069 specification. These data types are **string**, **int**, **unsignedInt**, **boolean**, **dateTime**, and **base64**, each of which is validated by a parameter dictionary (see Table 7-1 for data type definitions).

Example 5-1 illustrates how you can configure values for a parameter:

Example 5-1 Configuring Parameter Values

```
<tc:Template
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tc="urn:com:cisco:bac:common-template"
xmlns="urn:com:cisco:bac:cwmp-template"
xsi:schemaLocation="urn:com:cisco:bac:common-template CommonTemplateConstructs.xsd">

 <Configuration>
  <ParameterDictionaries>
    <ParameterDictionary>tr069-cwmp-dictionary.xml</ParameterDictionary>
  </ParameterDictionaries>
  <ObjectInstance name="InternetGatewayDevice">
    <ObjectInstance name="ManagementServer">
      <Parameter>
        <Name>PeriodicInformEnable</Name>
        <Value>true</Value>
      </Parameter>
      <Parameter>
        <Name>PeriodicInformInterval</Name>
        <Value>86400</Value>
      </Parameter>
    </ObjectInstance>
  </ObjectInstance>
 </Configuration>
</tc:Template>
```

Parameter List for Single Instance Object

This is an example of a single instance, or singleton:

`InternetGatewayDevice.UserInterface.`

The parameters of a singleton can be represented as:

```
InternetGatewayDevice.UserInterface.PasswordRequired = true
InternetGatewayDevice.UserInterface.ISPName = SBC
```

Parameter List for Multiple Instance Objects

The TR-069 specification defines some objects that can have multiple instances, with each instance assigned a number. You use the parameter dictionary to define the objects that can have multiple instances.

For example, `InternetGatewayDevice.Layer3Forwarding.Forwarding` is a multiple instance object.

Two instances of that object can be represented as described:

```
InternetGatewayDevice.Layer3Forwarding.Forwarding.1.Enable = true
InternetGatewayDevice.Layer3Forwarding.Forwarding.2.Enable = false
```

Note Instance numbers are generated by the device. A parameter dictionary uses `{i}` to represent the objects which can have multiple instances. Configuration templates must use the actual instance number to indicate the object which is being modified, such as `InternetGatewayDevice.WANDevice.1.WANConnectionNumberOfEntries.`

Notification

By using the Cisco BAC Notification attribute, a device can notify the DPE of parameter value changes. Notifications, by default, are turned off.

If Notification is enabled, it specifies that a device should include the changed values of a particular parameter in the device Inform message to the DPE, and that the device must initiate a session to the DPE whenever the value of that particular parameter is changed.

Cisco BAC supports the following values for the Notification attribute as defined in the TR-069 specification:

- `Off`—Notification set to off.

  The device need not inform Cisco BAC of a change to the specified parameter(s).

- `Passive`—Notification set to Passive.

  Whenever the specified parameter value changes, the device must include the new value in the ParameterList in the Inform message that is sent the next time that Cisco BAC establishes a session.

- `Active`—Notification set to Active.

  Whenever the specified parameter value changes, the device must initiate a session with Cisco BAC, and include the new value in the ParameterList in the associated Inform message. Whenever a parameter change is sent in the Inform message due to a nonzero Notification setting, the Event code `4 VALUE CHANGE` must be included in the list of Events.

Note The device returns a `notification request rejected` error if an attempt is made to set Notification on a parameter deemed inappropriate; for example, a continuously varying statistic.

Configuring Notification

The following example illustrates how you can configure the Notification attribute.

In this example, a configuration file is defined and generated for a `UserInterface` object, with Notification set to `Active` for all parameters. The `ISPName` parameter overrides this Notification and sets it to `Passive`.

```
<!--Set Notification of object InternetGatewayDevice.UserInterface.-->

<ObjectInstance name="InternetGatewayDevice">
    <ObjectInstance name="UserInterface">
            <Notification>Active</Notification> <!-- applies to all parameters under this
                                                     object -->
            <Parameter>
                <Name>PasswordRequired</Name>
                <Value>true</Value>
            </Parameter>
            <Parameter>
                <Name>WarrantyDate</Name>
                <Value>2001-02-23T09:45:30+04:30</Value>
            </Parameter>
            <Parameter>
                <Name>ISPName</Name>
                <Value>SBC</Value>
                <Notification>Passive</Notification> <!--applies to only this parameter-->
            </Parameter>
            <Parameter>
                <Name>ISPHomePage</Name>
```

```
                <Value>www.sbc.com</Value>
            </Parameter>
    </ObjectInstance>
</ObjectInstance>
```

Access Control

Access control in Cisco BAC is enabled through the *AccessControl* attribute of the CPE that you can control by using configuration templates. CPE parameters can be changed using LAN autoconfiguration protocol if access control is set to allow such access. For instance, if the *AccessControl* attribute is set to `Subscriber`, then the subscriber on the LAN can change the parameter value.

The value of the *AccessControl* attribute is an array of zero or more entities for which `write` access to a specific parameter is granted. If no entries are present, only updates to the parameter value are allowed from Cisco BAC.

Cisco BAC defines only one entity in this list, in line with the TR-069 specification: Subscriber. This enables `write` access by the subscriber on the LAN, such as through the LAN-side DSL CPE Configuration protocol or through the UPnP protocol.

Configuring Access Control

The following example illustrates how to set access control in Cisco BAC by using configuration templates:

```
    <ObjectInstance name="InternetGatewayDevice">
            <ObjectInstance name="ManagementServer">
                <AccessControl>  <!--Allow LAN-side updates of this object and parameters
under it -->
                        <Entity>Subscriber</Entity>
                </AccessControl>
                <Parameter>
                    <Name>PeriodicInformEnable</Name>
                    <Value>true</Value>
                </Parameter>

                <Parameter>
                    <Name>PeriodicInformEnable</Name>
                    <AccessControl/>   <!-- Allow updates only from ACS (BAC) -->
                </Parameter>
            </ObjectInstance>
        </ObjectInstance>
```

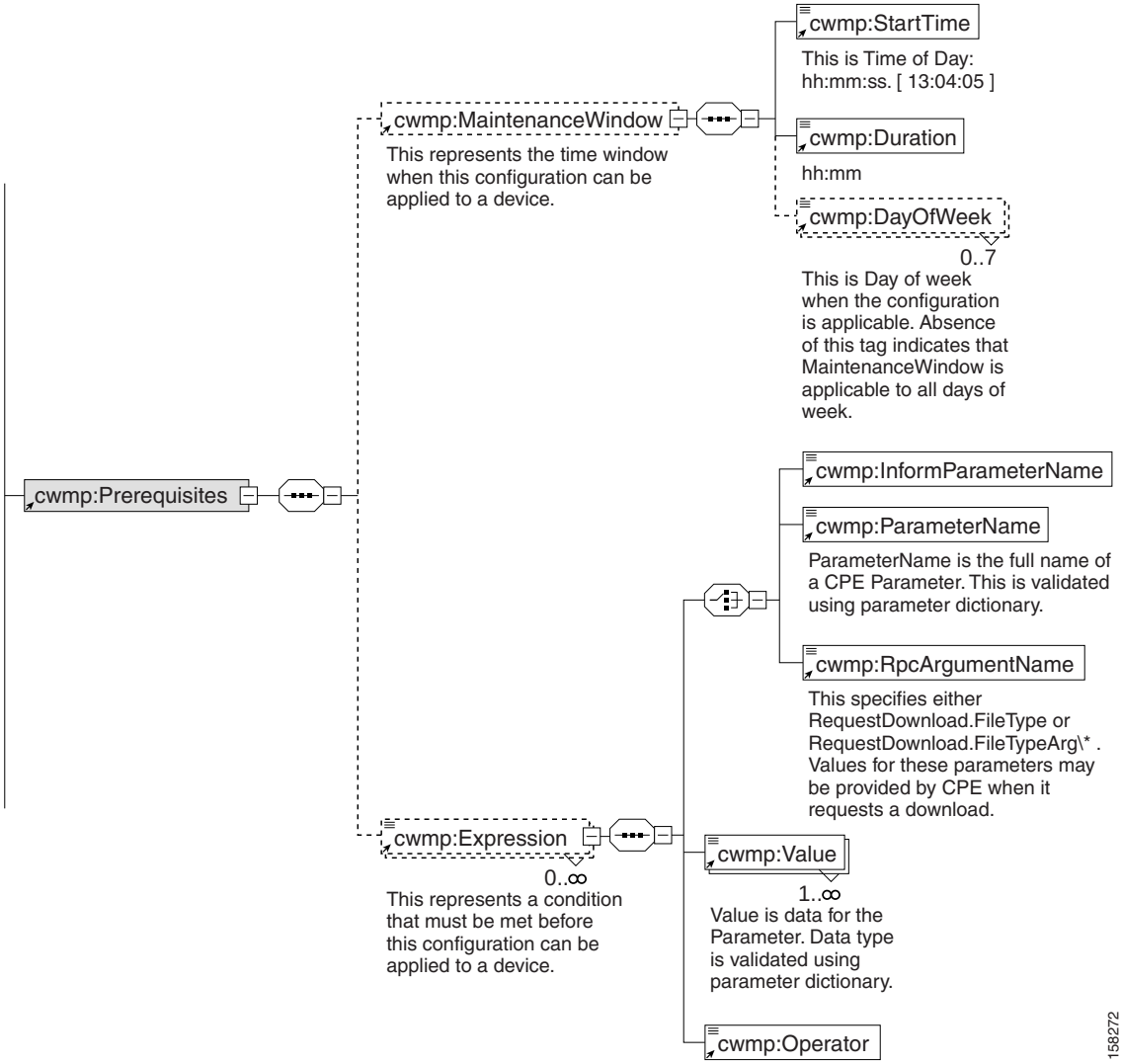
Prerequisites

Cisco BAC templates include a prerequisites option, which indicates what conditions must be met before configuring the device. For example, a prerequisite may specify that the device to which you apply a configuration must meet the requirement of a specific hardware or software version.

Prerequisite rules are embedded into an instruction, which is sent to the DPE. The DPE then interrogates the device as necessary to determine if the prerequisites match.

Figure 5-3 illustrates the schema of the prerequisites option:

Figure 5-3 Prerequisite Schema



You use prerequisites to manipulate configuration generation through:

- Expressions
- MaintenanceWindow

Expressions

Expressions are conditionals that use information from device properties. They may use Parameters, InformParameters or rpcArguments. Expressions are used within prerequisites and firmware rules and are processed when a device contacts Cisco BAC to retrieve its configuration.

If the expressions that you specify evaluates to *true*, the condition(s) is met, and the corresponding configuration or firmware rule is applied to the device. For detailed information, see Using Conditionals, page 5-19.

The *Prerequisites* tag may contain zero or more expressions.

SendHttpErrorOnFail

When the prerequisites specified in the configuration template are not met, and the sendHttpErrorOnFail attribute is set, an HTTP error with the specified code is sent to the device, and the corresponding configuration is not applied to the device.

Example 5-2 Configuring Prerequisite - sendHttpErrorOnFail

In this example, when the software version of the device is not 1.1, an HTTP error code is sent to the device.

```
<Prerequisites sendHttpErrorOnFail="503">
    <Expression>
        <ParameterName>InternetGatewayDevice.DeviceInfo.SoftwareVersion</ParameterName>
        <Value>1.1</Value>
        <Operator>match</Operator>
    </Expression>
</Prerequisites>
```

MaintenanceWindow

Use the MaintenanceWindow setting to specify the time window within which you want to apply a particular configuration to a specific CPE. This evaluation is performed at the DPE on device contact; as such, the DPE local time is used.

Note You should configure the server on which the DPE runs with automatic time server synchronization, such as NTP.

MaintenanceWindow comprises:

- *StartTime*—Specifies a time value indicating the beginning of a maintenance window. This value is defined as *hh:mm:ss*.
- *Duration*—Specifies a time value indicating the duration from *StartTime* when the configuration can be applied. This value is defined as *hh:mm*.
- *DayOfWeek*—Specifies the days of the week when you can apply the configuration. Absence of this tag indicates that the rule is valid for all days of the week. This tag is optional.

You can use these elements to specify the time window when this rule can run, thus providing the ability to limit configuration upgrades to late in the night when subscribers are most likely sleeping.

The *Prerequisites* tag may contain zero or one MaintenanceWindow.

Example 5-3 Configuring MaintenanceWindow

In the following example, the values of the *StartTime*, *Duration,* and *DayOfWeek* tags define the period when this rule is in effect. This template takes effect at 1 a.m. for a period of 5 hours, on every Monday, Tuesday, and Friday.

```
<Prerequisites>
    <MaintenanceWindow>
        <StartTime>01:00:00</StartTime>
        <Duration>05:00</Duration>
        <DayOfWeek>Monday</DayOfWeek>
        <DayOfWeek>Tuesday</DayOfWeek>
        <DayOfWeek>Friday</DayOfWeek>
    </MaintenanceWindow>
<Prerequisites>
```

Device Contact During MaintenanceWindow

A key concept in MaintenanceWindow is ensuring that devices contact Cisco BAC during the window.

You can configure the devices to contact the DPE at a given date and time, by setting a value for the *PeriodicInformTime* variable in the configuration template. Also, set the *RandomDateTimeInRange* tag to designate a time range when the device can contact Cisco BAC.

See Example 5-4 for how to configure a device to contact Cisco BAC during a MaintenanceWindow.

Example 5-4 Configuring Device to Contact Cisco BAC During MaintenanceWindow

In the following example, the device contacts Cisco BAC for the first time on January 1, 2007, between 3 a.m. and 4 a.m. After the initial contact, the device continues to contact Cisco BAC every 30 minutes.

- `PeriodicInformEnable` is set to *true* to indicate that the device must periodically send device information to Cisco BAC using the Inform method call.
- `PeriodicInformInterval` is set to 30 minutes (1800 seconds), requesting the device to attempt to contact Cisco BAC every 30 minutes. This duration is the interval within which the device attempts to connect with Cisco BAC if `PeriodicInformEnable` is *true*.
- `PeriodicInformTime` is set to a date and time when the device should initiate contact with Cisco BAC.

```
<ObjectInstance name="InternetGatewayDevice">
    <ObjectInstance name="ManagementServer">
            <Parameter>
                <Name>PeriodicInformEnable</Name>
                <Value>true</Value>
            </Parameter>
            <Parameter>
                <Name>PeriodicInformInterval</Name>
                <Value>1800</Value>
            </Parameter>
            <Parameter>
                <Name>PeriodicInformTime</Name>
                <ComplexValue>
                    <RandomDateTimeInRange>
                    <StartDateTime>2006-01-27T01:00:00+03:00</StartDateTime>
                    <RandomizationIntervalMinutes>60</RandomizationIntervalMinutes>
                    </RandomDateTimeInRange>
                </ComplexValue>
            </Parameter>
    </ObjectInstance>
</ObjectInstance>
```

For detailed information on PeriodicInformInterval, see the Broadband Forum's Technical Report on TR-069.

You can initiate a firmware upgrade in a similar fashion. Firmware upgrades are typically carried out only within a specific time range. After defining a firmware rule template with the MaintenanceWindow option set, associate the firmware rule template to a Class of Service object by using the administrator user interface.

After you finish this task, the firmware rule is pushed to the DPE. The DPE evaluates the firmware rule and ensures that the firmware upgrade is executed during the specific time.

Configuring Prerequisites

This section illustrates how to configure the prerequisite option.

Example 5-5 Configuring Prerequisites - Maintenance Window

In this example, the prerequisite indicates that this template will come into effect at 1 a.m. for a duration of 5 hours. During this time, this template can be applied to any device that contacts Cisco BAC and has an `EventCode` of *1 BOOT* and the manufacturer as *Acme, Inc*.

Example 5-6 Prerequisite Using Expression

```
<Prerequisites>
   <MaintenanceWindow>
     <StartTime>01:00:00</StartTime>
     <Duration>5:00</Duration>
   </MaintenanceWindow>
   <Expression>
     <InformParameterName>InternetGatewayDevice.DeviceInfo.EventCode</InformParameterName>
     <Value>1 BOOT</Value>
     <Operator>match</Operator>
   </Expression>
   <Expression>
     <ParameterName>InternetGatewayDevice.DeviceInfo.Manufacturer</ParameterName>
     <Value>Acme, Inc</Value>
     <Operator>matchIgnoreCase</Operator>
    </Expression>
 </Prerequisites>
```

Example 5-7 Prerequisite Using Regular Expression (Regex)

```
<Prerequisites>
   <MaintenanceWindow>
     <StartTime>01:00:00</StartTime>
     <Duration>5:00</Duration>
   </MaintenanceWindow>
   <Expression>
    <InformParameterName>Inform.EventCode</InformParameterName>
          <Value>^[0-9]\s\w</Value>
          <Operator>regexMatch</Operator>
   </Expression>
</Prerequisites>
```

Authoring Configuration Templates

You can use Cisco BAC to generate instructions for customized configurations for many devices from a single template by using template constructs. Parameter substitution and conditional inclusion or exclusion of template content, controlled by values from the Cisco BAC property hierarchy, create a custom configuration from a template. A template may include other templates, providing a mechanism to reuse common configurations.

You must add template files to the RDU by using the administrator user interface or the API, before any Class of Service can reference it.

Note The XML elements in a template are divided into two groups:

- Elements that are not prefixed by **tc** are unique to configuration templates.
- Elements prefixed by **tc** are generic constructs that are the same for configuration templates and for firmware rule templates.

A configuration template, with `Configuration` as the root element, must follow a structure similar to Example 5-8.

Example 5-8 Sample Configuration Template

```
<tc:Template
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tc="urn:com:cisco:bac:common-template"
xmlns="urn:com:cisco:bac:cwmp-template"
xsi:schemaLocation="urn:com:cisco:bac:common-template CommonTemplateConstructs.xsd">
<Prerequisites>
    <MaintenanceWindow>
        <StartTime>01:00:00</StartTime>
        <Duration>2:30</Duration>
    </MaintenanceWindow>
    <Expression>
        <ParameterName>InternetGatewayDevice.DeviceInfo.Manufacturer</ParameterName>
        <Value>Acme, Inc</Value>
        <Operator>matchIgnoreCase</Operator>
    </Expression>
 </Prerequisites>
 <Configuration>
  <ParameterDictionaries>
    <ParameterDictionary>tr069-cwmp-dictionary.xml</ParameterDictionary>
  </ParameterDictionaries>
  <ObjectInstance name="InternetGatewayDevice">
    <ObjectInstance name="ManagementServer">
      <Parameter>
        <Name>PeriodicInformEnable</Name>
        <Value>true</Value>
      </Parameter>
      <Parameter>
        <Name>PeriodicInformInterval</Name>
        <Value>86400</Value>
      </Parameter>
    </ObjectInstance>
  </ObjectInstance>
 </Configuration>
</tc:Template>
```

Configuration templates support:

- Parameter substitution—Substitutes values from the Cisco BAC property hierarchy into XML element content and element attributes by using the `VAR()` construct. (For details, see Using Parameter Substitution, page 5-16).
- Includes—Build a set of reusable template snippets. Includes are useful for defining options that are common across many service classes without having to duplicate the options in several templates. (For details, see Using Includes, page 5-17.)
- Conditionals—Include or exclude blocks of text within a template. The text blocks that you can enclose within the conditional statements are limited to parameter and object instance elements. (For details, see Using Conditionals, page 5-19.)

Custom Properties

Cisco BAC properties provide access to data that is stored in Cisco BAC using the API. Using the API, you can use the properties of corresponding objects to retrieve preprovisioned, discovered, and status data. You can use properties to configure Cisco BAC at the appropriate level of granularity (from system level to device group and to individual device). See Property Hierarchy, page 4-4 for details.

You can use custom properties to define additional customizable device information to be stored in the RDU database. You typically use these properties to substitute parameter values in configuration templates and firmware rule templates.

The template parser works from the bottom up while locating properties in the hierarchy (device first, followed by the Group, provisioning group, then the Class of Service, device type, and system defaults) and converts the template option syntax.

For details on substituting template parameters using the `VAR()` construct, see the subsequent section.

Example 5-9 Retrieving Username or Password from Device

The following example illustrates how you can use templates to retrieve properties from different levels; for instance, username and password credentials from a device.

```
<tc:Template>
    <Configuration>
        <ParameterDictionaries>
            <ParameterDictionary>tr069-cwmp-dictionary.xml</ParameterDictionary>
        </ParameterDictionaries>
        <ObjectInstance name="InternetGatewayDevice">
            <ObjectInstance name="ManagementServer">
                <Parameter>
                    <Name>PeriodicInformEnable</Name>
                    <Value>true</Value>
                </Parameter>
                <Parameter>
                    <Name>PeriodicInformInterval</Name>
                    <Value>20</Value>
                    <Notification>Active</Notification>
                </Parameter>
                <Parameter>
                    <Name>ConnectionRequestUsername</Name>
                    <Value>VAR(name=/dt/username, defaultValue="User")</Value>
                </Parameter>
                <Parameter>
                    <Name>ConnectionRequestURL</Name>
                    <Value>VAR(name=/dt/pk, defaultValue="http://testURL")</Value>
                </Parameter>
```

```
            </ObjectInstance>
        </ObjectInstance>
    </Configuration>
</tc:Template>
```

To configure custom properties from the administrator user interface, choose the **Configuration > Custom Property** tabs. Use the Add Custom Property page to add or delete custom properties. For details, see Configuring Custom Properties, page 17-5.

Caution If you remove custom properties that a template references, the RDU fails to generate instructions for a device.

Using Parameter Substitution

Values from the Cisco BAC property hierarchy are substituted into a template by using the `VAR()` construct, to produce a custom configuration from a configuration template. The `VAR()` construct can appear in an XML element value or element attribute. You can also use it to substitute full or partial values.

The following list describes the constructs that Cisco BAC supports for parameter substitution:

- Cisco BAC property value into XML element content
- Cisco BAC property value into XML element attribute
- Default value
- XML partial element content
- Values with special characters

Syntax Description

`VAR(`*`token`*`=someChar, `*`name`*`=someProperty, `*`defaultValue`*`=someValue)`

- *token*—Specifies the character that delimits subsequent fields. This element is optional and defaults to a comma (,).

  Note If the default value contains a comma (,), then start the `VAR()` construct by specifying a token character. This token must not appear in the default value. Also, ensure that the token character precedes the `VAR` end bracket.

- *name*—Specifies the name of the custom property to be substituted or the Standard Device properties to be referenced.
- *defaultValue*—Specifies the value to use if the referenced property is not available.

Example 5-10 Setting a Value to a Cisco BAC Custom Property

```
<Value>VAR(name=/cpe/version, defaultValue=4)</Value>
```

Example 5-11 Referencing Standard Device Property

```
<Value>VAR(name=/IPDevice/connectionRequestPath, defaultValue="http://test")</Value>
```

Note The API constant for /IPDevice/connectionRequestPath is `IPDeviceKeys.CONNECTION_REQUEST_PATH`.

Example 5-12 Using a `defaultValue` With a Comma

```
<Value>VAR(token=;;name=/cpe/usrStr; defaultValue=4,5;)</Value>
```

Example 5-13 Creating a String Value based on a Cisco BAC Property

```
<Value>CWMP_VAR(name=/cpe/version, defaultValue=4).bin</Value>
```

If `cpe/version` is `1-08` then the Value will be `CWMP_1-08.bin`.

Using Includes

Include files let you build a set of reusable template snippets. These files are useful for defining options that are common across many service classes without having to duplicate the options in several templates.

You can include the content of a particular file into a template by using the **tc:include** construct. After inserting the content of included files into the host template, the parameter dictionary that is specified in the host template validates the content of the resulting template.

Note If included templates use objects and parameters not defined in the same dictionary as the host template, parameter validation fails during instruction generation.

The **tc:Include** element specifies the *href* attribute, where *href* identifies the name of the Cisco BAC template file that is included in the host template. Use double quotation marks (") when using an include directive in template.

Note Ensure that Include files are of the same file type as the template. For instance, when adding a configuration template, all the included files must be of the Configuration Template file type.

Example 5-14 Include Files

The following example displays the content of a host template, and an included template.

Included Template: *informInterval.xml*

```
<!-- enable and set Periodic Inform value -->
<tc:Template :xsi="http://www.w3.org/2001/XMLSchema-instance"
:tc="urn:com:cisco:bac:common-template" ="urn:com:cisco:bac:cwmp-template"
:schemaLocation="urn:com:cisco:bac:common-template CommonTemplateConstructs.xsd">
    <Configuration>
        <ParameterDictionaries>
            <ParameterDictionary>tr069-cwmp-dictionary.xml</ParameterDictionary>
        </ParameterDictionaries>
        <ObjectInstance name="InternetGatewayDevice">
            <ObjectInstance name="ManagementServer">
                <Parameter>
                    <Name>PeriodicInformEnable</Name>
                    <Value>true</Value>
```

```
                </Parameter>
                <Parameter>
                    <Name>PeriodicInformInterval</Name>
                    <Value>30</Value>
                </Parameter>
            </ObjectInstance>
        </ObjectInstance>
    </Configuration>
</tc:Template>
```

Host Template: *cwmp-config.xml*

```
<!-- set Periodic Inform value based on content of informInterval-cwmp.xml -->
<!-- set ManagmentServer.URL -->
<tc:Template :xsi="http://www.w3.org/2001/XMLSchema-instance"
:tc="urn:com:cisco:bac:common-template" ="urn:com:cisco:bac:cwmp-template"
:schemaLocation="urn:com:cisco:bac:common-template CommonTemplateConstructs.xsd">
<Configuration>
        <ParameterDictionaries>
            <ParameterDictionary>tr069-cwmp-dictionary.xml</ParameterDictionary>
        </ParameterDictionaries>
        <tc:include href="informInterval.xml"/>
        <ObjectInstance name="InternetGatewayDevice">
            <ObjectInstance name="ManagementServer">
                <Parameter>
                    <Name>URL</Name>
                    <Value>http://10.44.64.200:9595/acs</Value>
                </Parameter>
            </ObjectInstance>
        </ObjectInstance>

    </Configuration>
</tc:Template>
```

Template After Inserting Included File: *informInterval.xml*

```
<!-- set ManagmentServer.URL -->
<tc:Template :xsi="http://www.w3.org/2001/XMLSchema-instance"
:tc="urn:com:cisco:bac:common-template" ="urn:com:cisco:bac:cwmp-template"
:schemaLocation="urn:com:cisco:bac:common-template CommonTemplateConstructs.xsd">
    <Configuration>
        <ParameterDictionaries>
            <ParameterDictionary>tr069-cwmp-dictionary.xml</ParameterDictionary>
        </ParameterDictionaries>
        <ObjectInstance name="InternetGatewayDevice">
            <ObjectInstance name="ManagementServer">
                <Parameter>
                    <Name>PeriodicInformEnable</Name>
                    <Value>true</Value>
                </Parameter>
                <Parameter>
                    <Name>PeriodicInformInterval</Name>
                    <Value>30</Value>
                </Parameter>
            </ObjectInstance>
        </ObjectInstance>
        <ObjectInstance name="InternetGatewayDevice">
            <ObjectInstance name="ManagementServer">
                <Parameter>
                    <Name>URL</Name>
                    <Value>http:// 10.44.64.200:9595/acs</Value>
                </Parameter>
            </ObjectInstance>
        </ObjectInstance>
```

```
    </Configuration>
</tc:Template>
```

Using Conditionals

Cisco BAC supports powerful conditional expressions in template constructs to provide ultimate configuration customization.

You can use conditional expression constructs to include or exclude blocks of text within a template. These construct elements are **tc:if**, **tc:choose**, and **tc:when**. You use the **tc:if** construct for simple single conditionals.

You can use a combination of **tc:choose**, **tc:when**, and **tc:otherwise** for a logical **if...else if...else** construct. The **tc:if** and the **tc:when** constructs require a *test* attribute. A *test* attribute is an expression that can be evaluated, and the resulting object can be converted to a Boolean operator (*true* or *false*).

The **tc:choose** element selects one from a number of possible alternatives. This element contains a sequence of **tc:when** elements, followed by an optional **tc:otherwise** element. Each **tc:when** element has a single attribute, *test*, which specifies an expression.

The content of the **tc:when** and the **tc:otherwise** elements is a valid template segment. When a **tc:choose** element is processed, each of the **tc:when** elements is tested in turn, by evaluating the expression and converting the resulting object to a Boolean operator.

The content of the first—and only the first—**tc:when** element whose test is true is instantiated. If no **tc:when** is true, the content of the **tc:otherwise** element is instantiated. If no **tc:when** element is true and no **tc:otherwise** element is present, nothing is created.

Note Within expressions, you can use single (‘) or double quotes (“) to delimit literal strings. The template processor interprets a quotation mark (‘ or “) as terminating the attribute. To avoid this problem, you can enter the marks as a character reference (" or '). (See Table 5-2.) Alternatively, the expression can use single quotes (‘) if the attribute is delimited with double quotes (“) or vice-versa.

Table 5-2 lists the conditional template constructs that this Cisco BAC release supports.

Table 5-2 Conditional Template Constructs Supported in Cisco BAC

Conditional statement	If
Conditional statement	If...else if...else
Conditional statement	or
Conditional expression	and
Conditional expression	lessThan
Conditional expression	lessThanEquals
Conditional expression	greaterThan
Conditional expression	greaterThanEquals
Conditional expression	contains, notContains
Conditional expression	containsIgnoreCase, notContainsIgnoreCase
Conditional expression	equals, notEquals
Conditional expression	equalsIgnoreCase, notEqualsIgnoreCase

Table 5-2 Conditional Template Constructs Supported in Cisco BAC (continued)

Conditional expression	when..otherwise
Conditional expression	choose

XML does not support the left angle bracket (<) and the ampersand (&). Instead, use the predefined entities (described in Table 5-3) for these characters.

Table 5-3 XML Definitions in Cisco BAC

Instead of ...	Use ...
< (less than)	<
<= (less than or equal)	<=
> (greater than)	>
>= (greater than or equals)	>=
& (ampersand)	&
' (apostrophe)	'
" (double quotes)	"

The following examples illustrate various conditional expressions:

Example 5-15 Numerical Test Condition

Using 100 as a number:

```
<tc:if test="VAR(name=/cpe/version, defaultValue=20) greaterThan 100">
    <Parameter>
        <Name>Enable</Name>
        <Value>false</Value>
    </Parameter>
</tc:if>
```

Example 5-16 String Test Condition

Using 100 as string:

```
<tc:if test="VAR(name=/cpe/version, defaultValue=20) greaterThan '100'">
    <Parameter>
        <Name>Enable</Name>
        <Value>false</Value>
    </Parameter>
</tc:if>
```

Note When comparing digits as numbers, 20 is less than 100. However, when an operator is used with a string, comparing digits as strings makes 20 greater than 100.

Example 5-17 Looking for Text in a Cisco BAC Property

Using text `cwmp`:

```
<tc:if test="contains(VAR(name=/cpe/UsrStr, defaultValue=linksys), 'cwmp')">
    <Parameter>
        <Name>Enable</Name>
        <Value>false</Value>
    </Parameter>
</tc:if>
```

Example 5-18 Looking for Text in Single Quotes

Using `cwmp's test`:

```
<tc:if test="contains(VAR(name=/cpe/UsrStr, defaultValue=linksys), &quot;cwmp's
test&quot;)">
    <Parameter>
        <Name>Enable</Name>
        <Value>false</Value>
    </Parameter>
</tc:if>
```

Example 5-19 Looking for Text in Double Quotes

Using `cwmp` *test*:

```
<tc:if test="contains(VAR(name=/cpe/UsrStr, defaultValue=linksys), 'cwmp
&quot;test&quot;')">
    <Parameter>
        <Name>Enable</Name>
        <Value>false</Value>
    </Parameter>
</tc:if>
```

The text blocks that you can enclose within the conditional statements are limited to `Parameter` and `Object` elements.

Example 5-20 Choosing from Alternatives (Using choose and when Syntax)

The following example describes the syntax when using **tc:choose** and **tc:when**:

```
<tc:choose>
        <tc:when test="VAR(name=/cpe/version,defaultValue=11) greaterThan 12">
            <Parameter>
                <Name>UpgradeAvailable</Name>
                <Value>true</Value>
            </Parameter>
        </tc:when>
        <tc:when test="VAR(name=/db/version,defaultValue=15) greaterThan 14">
            <Parameter>
                <Name>UpgradeAvailable</Name>
                <Value>true</Value>
            </Parameter>
        </tc:when>
        <tc:otherwise>
            <Parameter>
                <Name>UpgradeAvailable</Name>
                <Value>false</Value>
            </Parameter>
        </tc:otherwise>
</tc:choose>
```

Example 5-21 Choosing from Alternatives (Using choose and if Syntax)

The following example describes the syntax when using **tc:choose** and **tc: if**:

```
<tc:choose>
        <tc:if test="contains(VAR(name=FC-ACTIVATED-INFORM-ENABLE, defaultValue=false),
'true')">
            <Parameter>
                <Name>PeriodicInformInterval</Name>
                <Value>21600</Value>
            </Parameter>
        </tc:if>
        <tc:else>
            <Parameter>
                <Name>PeriodicInformInterval</Name>
                <Value>1200</Value>
            </Parameter>
        </tc:else>
</tc:choose>
```

Using the Configuration Utility

You can use the configuration utility to test, validate, and view TR-069 template files: configuration and firmware rules. These activities are critical to successful deployment of instructions for individualized configuration files. (For more information on templates, see Authoring Configuration Templates, page 5-14.)

The configuration utility is available only when the RDU is installed, and it is installed in the <BPR_HOME>/rdu/bin directory.

All examples in this section assume that the RDU is operating and that these conditions apply:

- The BAC application is installed in the default home directory (*/opt/CSCObac*).
- The RDU login name is **bacadmin**.
- The RDU login password is **changeme**.

Note Some of the examples provided in this section were truncated whenever the omitted formation is of no consequence to the example of its outcome. Instances where this occurs are identified by an ellipsis (...) that precede the example summary.

This section discusses:

- Running the Configuration Utility, page 5-23
- Adding a Template to Cisco BAC, page 5-23
- Validating XML Syntax for a Local Template File, page 5-24
- Validating XML Syntax for a Template Stored in Cisco BAC, page 5-24
- Testing Template Processing for a Local Template File, page 5-25
- Testing Template Processing for a Template Stored in Cisco BAC, page 5-26
- Testing Template Processing for a Cisco BAC Template File and a Device, page 5-27

Running the Configuration Utility

In subsequent procedures and examples, the phrase running the configuration utility means to enter the **runCfgUtil.sh** command from the directory specified. To run the configuration utility, run this command from the *<BPR_HOME>/rdu/bin* directory:

```
runCfgUtil.sh options
```

The available *options* include:

- **-cwmp**—Specifies the input file is to be processed as a CWMP technology configuration template file. Use -cwmp with the -a option.
- **-a** {**sc** | **gc**}—Specifies a required action, which could be either:
  - **sc**—To specify a syntax check to test if the XML template and dictionary are well-formed. Use **sc** with the **–l** option.
  - **gc**—To process a configuration template the same way as the Instruction Generation Service (IGS) would. Use **gc** with the (**–l** and **–data**), or the (**-l** and **-i**) options.
- **-l** *filename*—Identifies the input file to be on the local file system. For example, if the name of your input file is *any_file*, enter **-l any_file**.

  Note Do not use **-l** with the **-r** option.

- **-o** *filename*—Saves the output of template processing in XML format into the specified file. For example, to save the output in a file called *op_file*, enter **-o op_file** This is an optional element.
- **-i** *device id*—Specifies the device to use for variable substitution when processing the template. Variable values are retrieved by using the device's properties. Use with the **-r** option.
- **-r** *filename*—Specifies name of template stored in the RDU. Use this option instead of the **-l** option.
- **-u** *username*—Specifies the username to be used when connecting to the RDU.
- **-p** *password*—Specifies the password to be used when connecting to the RDU.
- **-firmware**—Specifies the input file to be a firmware rule template. Otherwise, assume configuration template.

Adding a Template to Cisco BAC

To use the configuration utility to test BAC templates:

Step 1 Develop the template as described in Authoring Configuration Templates, page 5-14. If the template includes other templates, make sure all the referenced templates are in the same directory.

Step 2 Run the configuration utility on the local file system. You can check the syntax for the template, or have the configuration utility process template as IGS would, and return XML output.

If the processed template uses `VAR()` constructs to reference a device property, the resulting output will contain the defaultValues for those `VAR()` constructs.

Step 3 Add the template (and any included templates that are used) to the RDU.

If the template uses `VAR()` constructs to reference a device property, use the device option to generate a sample template.

Step 4 After all tests succeed, configure a Class of Service to use the template.

Validating XML Syntax for a Local Template File

Use the **runCfgUtil.sh** command to validate template files stored on the local file system.

Syntax Description

`runCfgUtil.sh -cwmp -a sc -l` *`file`*

- **-cwmp**—Identifies the input file to be processed as a CWMP template file.
- **-a sc**—Specifies a syntax check.
- **-l**—Specifies that the input file is on the local file system.
- *file*—Identifies the input template file being validated.

To use a template file that is on the local file system:

Step 1 Change directory to */opt/CSCObac/rdu/samples/cwmp*.

Step 2 Select a template file to use.

Note This example uses an existing template file called *sample-cwmp-config.xml*. The **-cwmp** option is used because this is a CWMP template.Run the configuration utility by using this command:

```
/opt/CSCObac/rdu/bin/runCfgUtil.sh -cwmp -a sc -l sample-cwmp-config.xml
```

- **-cwmp**—Identifies the input file as a CWMP template file.
- **-l**—Specifies that the input file is on the local file system.
- **sample-cwmp-config.xml**—Identifies the input template file being validated.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 3.0, Revision: 1.26
validating configuration template sample-cwmp-config.xml...
>sample-cwmp-config.xml is valid.
```

Validating XML Syntax for a Template Stored in Cisco BAC

Use the **runCfgUtil.sh** command to validate template files that are in the RDU database.

Syntax Description

`runCfgUtil.sh -cwmp -a sc -r` *`file`* **`-u`** *`username`* **`-p`** *`password`*

- **-cwmp**—Identifies the input file to be processed as a CWMP template file.
- **-a sc**—Specifies a syntax check.
- **-r** *file*—Identifies the input file as a file that has been added to the RDU.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*—Specifies the password to use when connecting to the RDU.

To validate a template file that has been added to the RDU:

Step 1 Change directory to */opt/CSCObac/rdu/samples/cwmp*.

Step 2 Select a template file to use.

Note This example uses an existing template file called *sample-cwmp-config.xml*. The **-cwmp** option is used because a CWMP template is being used.

Step 3 Run the configuration utility by using this command:

```
./runCfgUtil.sh -cwmp -a sc -r sample-cwmp-config.xml -u admin -p changeme
```

- **sample-cwmp-config.xml**—Identifies the input file.
- **admin**—Identifies the username.
- **changeme**—Identifies the password.
- **-cwmp**—Identifies the file as a CWMP template.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 3.0, Revision: 1.26
validating configuration template sample-cwmp-config.xml...
>sample-cwmp-config.xml is valid.
```

Testing Template Processing for a Local Template File

Use the **runCfgUtil.sh** command to test template processing for a local template file.

Syntax Description

```
runCfgUtil.sh -cwmp -a gc -l file -o file
```

- **-cwmp**—Identifies the input file to be processed as a CWMP template file.
- **-a gc**—Specifies instructions for configuration generation.
- **-l** *file*—Specifies that the input file is on the local file system.
- **-o** *file*—Specifies that the processed template be saved in XML format into the specified file.

To test template processing for a local template file:

Step 1 Change directory to */opt/CSCObac/rdu/samples/cwmp*.

Step 2 Select a template file to use. This example uses the existing template file, *sample-cwmp-config.xml*.

Step 3 Run the configuration utility by using this command:

```
./opt/CSCObac/rdu/bin/runCfgUtil.sh -cwmp -a gc -l sample-cwmp-config.xml -o output.xml
```

- **sample-cwmp-config.xml**—Identifies the input file.
- **output.xml**—Identifies the file in which to save the processed template in XML format.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 3.0, Revision: 1.26
generating configuration from sample-cwmp-config.xml...
output.xml generated successfully.
```

You can view the configuration from the *output.xml* file.

Testing Template Processing for a Template Stored in Cisco BAC

Use the **runCfgUtil.sh** command to test the template processing for a template in the RDU database.

Syntax Description

runCfgUtil.sh -cwmp -a gc -r *file* **-o** *file* **-u** *username* **-p** *password*

- **-cwmp**—Identifies the input file to be processed as a CWMP template file.
- **-a gc**—Specifies instructions for configuration generation.
- **-r** *file*—Identifies the input file as a file that has been added to the RDU.
- **-o** *file*—Specifies that the processed template be saved in XML format into the specified file.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*—Specifies the password to use when connecting to the RDU.

To test the template processing for a template stored in Cisco BAC:

Step 1 Change directory to */opt/CSCObac/rdu/samples/cwmp*.

Step 2 Select a template file to use. This example uses the existing template file, *sample-cwmp-config.xml*.

Step 3 Run the configuration utility by using this command:

```
./runCfgUtil.sh -cwmp -a sc -r sample-cwmp-config.xml -o output.xml -u admin -p changeme
```

- **sample-cwmp-config.xml**—Identifies the input file.
- **output.xml**—Identifies the file in which the generated configuration is to be saved in XML format.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 3.0, Revision: 1.26
generating configuration from sample-cwmp-config.xml...
output.xml generated successfully.
```

You can view the configuration from the *output.xml* file.

Testing Template Processing for a Cisco BAC Template File and a Device

Use the **runCfgUtil.sh** command to test the template processing for a file stored in the RDU database and associated with a device.

Syntax Description

runCfgUtil.sh -cwmp -a gc -r *file* **-o** *file* **-i** *deviceID* **-u** *username* **-p** *password*

- **-cwmp**—Identifies the input file to be processed as a CWMP template file.
- **-a gc**—Specifies instructions for configuration generation.
- **-r** *file*—Identifies the input file as a file that has been added to the RDU.
- **-o** *file*—Specifies that the processed template be saved in XML format into the specified file.
- **-i** *deviceID*—Specifies the device to use. Variable values are retrieved using the device properties.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*—Specifies the password to use when connecting to the RDU.

To test the template processing for a file stored in the RDU and associated with a device:

Step 1 By using the administrator user interface, add the template file */opt/CSCObac/rdu/samples/cwmp/sample-cwmp-var-config.xml* to Cisco BAC.

Note The *sample-cwmp-var-config.xml* template has a reference to the /IPDevice/connectionRequestUsername device property. The API constant for the property is `IPDeviceKeys.CONNECTION_REQUEST_USERNAME`.

Step 2 Select a template file to use. This example uses the existing template file, *sample-cwmp-var-config.xml*.

Step 3 Identify a device that is already in Cisco BAC and use that device's ID. This example uses a device with the /IPDevice/connectionRequestUsername property set to *testUser*.

Step 4 Identify the device to use. This example assumes that the device exists in the RDU and has the variables set as properties.

Step 5 Run the configuration utility by using this command:

```
./runCfgUtil.sh -cwmp -a gc -r sample-cwmp-var-config.xml -i OUI-1234 -o output.xml -u
admin -p changeme
```

- **sample-cwmp-var-config.xml**—Identifies the input file.
- **OUI-1234**—Identifies the ID of the device. The Device ID used here is for example purposes only.
- **output.xml**—Identifies the file in which to save the processed template in XML format.
- **bacadmin**—Identifies the default username being used in this example.
- **changeme**—Identifies the default password being used in this example

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 3.0, Revision: 1.26
generating configuration from sample-cwmp-var-config.xml...
output.xml generated successfully.
```

Note You can open the *output.xml* file to view the configuration. The `VAR(name=/IPDevice/connectionRequestUsername, defaultValue=test)` statement will be replaced with *testUser*.