Templates

When creating a template, it is possible to specify variables that will be contextually substituted. Many of these variables are available in the drop-down menu in the Template Editor (see Figure 12-4). It is also possible to create these files offline without the Template Editor and still use these variables.

The basic format of a template file is simply the text of the configuration to be downloaded to your device (see "Sample Template" section on page 12-1). However, you can put variable substitutions of the following form (for example, the variable name could be *iosipaddress*):

```
Internal directory mode:
   ${LDAP://this:attrName=iosipaddress}
External directory mode:
   ${LDAP://10.1.2.3/cn=Device1,ou=CNSDevices,o=cisco,c=us:attrName=iosipaddress}
```

It is possible to create segments of templates that can be included in other templates. For example, you might have an Ethernet configuration that would be used by multiple devices. In each device template, you could have:

```
#include /opt/CSCOcnsie/Templates/ethernet_setup.cfgtpl
```

Now, you could centralize all the administration for Ethernet configuration in one file.

Caution Circular includes of template files are not allowed.

Sample Template

The following sample is the configuration template for the DemoRouter (*DemoRouter.cfgtpl*), which is pre-loaded on your system:

```
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname DemoRouter
!
boot system flash c7200-is-mz
enable secret 5 $1$cMdI$.e37TH540MWB2GW5gMOn3/
enable password cisco
```

```
!
ip subnet-zero
!
interface FastEthernet0/0
 no ip address
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
 shutdown
 half-duplex
!
interface Ethernet1/0
 ip address 10.10.1.1 255.255.255.240
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
!
interface Ethernet1/1
 no ip address
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
 shutdown
!
interface Ethernet1/2
 no ip address
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
 shutdown
!
interface Ethernet1/3
 no ip address
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
 shutdown
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.10.1.1
ip http server
!
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password cisco
 login
!
end
```

Configuration Control Templates

To restart a device with a new image, you need Configuration Control templates that contain the required CLI commands for image activation on particular devices.

For example, if you want to restart a Cisco 3600 Series router with an image named *3600.image*, from the device console, you would issue the following CLI commands:

no boot system
boot system flash:3600.image

The content of the Configuration Control template for image activation should contain the CLI commands that you would normally enter from the device console to activate a new image on the device.

Dynamic Flow Control Template

The inventory information collected from image agents is made available for external users by means of the Dynamic Flow Control Template. This enables you to write templates that can control the flow of configuration and image distribution jobs, based on the inventory information.

Inventory Operations

These are the operations that are exposed to you to access the inventory of the device from the Dynamic Flow Control Templates:

Function	**$!{invObj.getDram()}**
Return Type	int (bytes).
Description	Dram = Main Mem Size + IO Mem Size. Returns the size of the DRAM.

Function	**$!{invObj.getVersionString()}**
Return Type	String.
Description	Returns the version string of the current running image from the device inventory.

Function	**$!{invObj.getImageFile()}**
Return Type	String.
Description	Returns the current running image file name.

Function	**$!{invObj.getImageMD5()}**
Return Type	String.
Description	Returns the MD5 as provided in the device inventory.

Function	**$!{invObj.getStartedAt()}**
Return Type	String.
Description	Returns the time string of when the device started.

Function	**$!{invObj.getPlatformName()}**
Return Type	String.
Description	Returns the platform name.

Function	**$!{invObj.getFlash()}**
Return Type	int (bytes).
Description	Returns the size of the flash.

Function	**$!{invObj.getFileSysSize("bootflash")}**
Return Type	int (bytes).
Description	Returns the size of the bootflash.

Function	**$!{invObj.getFileSysFreespace("bootflash")}**
Return Type	int (bytes).
Description	Returns the amount of free space in the bootflash.

Function	**$!{invObj.getFileSysSize("nvram")}**
Return Type	int (bytes).
Description	Returns the size of the NVRAM.

Function	**$!{invObj.getFileSysFreespace("nvram")}**
Return Type	int (bytes).
Description	Returns the amount of free space in the NVRAM.

Function	**$!{invObj.getFileSysSize("disk0")}**
Return Type	int (bytes).
Description	Returns the size of disk0.

Function	**$!{invObj.getFileSysFreespace("disk0")}**
Return Type	int (bytes).
Description	Returns the amount of free space in disk0.

Function	**$!{invObj.getFileSysSize("slot0")}**
Return Type	int (bytes).
Description	Returns the size of slot0.

Function	**$!{invObj.getFileSysFreespace("slot0")}**
Return Type	int (bytes).
Description	Returns the amount of free space in slot0.

Function	**$!{invObj.getFileSysSize("slot1")}**
Return Type	int (bytes).
Description	Returns the size of slot1.

Function	**$!{invObj.getFileSysFreespace("slot1")}**
Return Type	int (bytes).
Description	Returns the amount of free space in slot1.

Other Operations

These are the operations that are exposed to you to perform an action based on the above criterion from the Dynamic Flow Control Template:

Function	**$!{cnsceObj.distribute()}**
Parameters	None.
Description	Perform image distribution. The pre-configured image is used.

Function	**$!{cnsceObj.activate("persist" | "nv_overwrite")}**
Parameters	Sets the config action: • "persist" – apply and save configuration to NVRAM. • "nv_overwrite" – overwrite NVRAM configuration.
Description	Performs image activation. The pre-configured image is used.

Function	**$!{cnsceObj.updateConfig(true | false, "write" | "persist" | "nv_overwrite")}**
Parameters	First parameter sets the syntax check: • true – syntax check is turned on. • false – syntax check is turned off. Second parameter is to set the config action: • "write" – apply to running configuration. • "persist" – apply and save configuration to NVRAM. • "nv_overwrite" – overwrite NVRAM configuration.
Description	Performs configuration update. The pre-configured template is used.

Notes

The **invObj.getDram()** operation returns the following:

Dram = Main Mem Size + IO Mem Size

Example

```
#set( $dram = $!{invObj.getDram()} )
##
#if ($dram > 6100)
  $!{cnsceObj.distribute()}
  $!{cnsceObj.activate("persist")}
#end
```

As seen in the example above, you can customize the flow of the job depending on the DRAM size.

When a custom job with the above inventory template is submitted, the device is queried for its inventory, and depending on the DRAM size, the decision is made if the image upgrade is to be performed or not. Hence when the above example inventory template is evaluated, if the DRAM size of the device is greater than 6100 bytes the image distribution and image activation will be performed.

Sample1

```
#set( $dram = $!{invObj.getDram()} )
#set( $flash = $!{invObj.getFlash()} )
##
#if ($dram > 64000000)
    $!{cnsceObj.distribute()}
    #if ( $flash > 48000000 )
       $!{cnsceObj.activate("persist")}
    #end
#end
```

Sample 2

```
#set( $disk0free = $!{invObj.getFileSysFreespace("disk0")} )
##
#if $disk0free > 3500000)
    $!{cnsceObj.distribute()}
     $!{cnsceObj.activate("persist")}
#end
```

Sample 3

```
#set( $flash = $!{invObj.getFlash()} )
##
#if ($flash > 65000000)
    $!{cnsceObj.updateConfig(true, "persist")}
#end
```

Templates for Modular Routers

The template mechanism for the devices has been enhanced to support modular routers. A modular router chassis includes slots in which you can install modules. You can install any module into any available slot in the chassis. Some modules like 2 Ethernet 2 WAN card slot module can in turn have sub slots to install interface cards or line cards. Device management has been extended to support subdevices representing line cards.

Additional attributes representing line card number, line card type, and subdevices have been added to the existing device object structure in the directory server in order to have the same structure to represent the main device or the subdevice.

Currently, card type is a string that maps to the product code of the network module. Since the EPROM data in the card stores part numbers only, not product codes, the part numbers are mapped to product codes. The user uses part numbers and the configuration server maps part numbers to product codes.

In the context of main device, the line card number and line card type fields make no sense and hence are set to NULL value. The subdevices field in the sub device (representing the line card) is set to NULL value.

New interface variable support has been added. These variables are included in the templates, which are parameterize with the interface numbers in the template. These are not attributes. They are special format variables that are replaced by the configuration server based on the interface information, which comes from the device. These variables only specify the relative position of the interface on the module and are replaced by the actual slot number, shelf-ID or port number. The interface variables are wrapped in percent sign (%) characters and specify the type, if any, and the relative position. The configuration server replaces these variables with the interface numbers. The interface type still has to be specified in the CLI using the following syntax:

Interface Variable = %[InterfaceType] RelativePosition%

For example:

%FastEthernet 0% for interface FastEthernet

%Serial 0% interface Serial

%T1 0% controller T1

%E1 0% controller E1

%voice-port 0% voice-port

Example 1:

A network module with two FastEthernet ports plugged in Slot 2 would be referred in the configuration CLI as FastEthernet 2/0 and FastEthernet 2/1 and referred in the template as FastEthernet %FastEthernet 0% and FastEthernet %FastEthernet 1%:

```
!
interface FatsEthernet 2/0
    ip address 10.10.1.1 255.255.255.0
!
interface FatsEthernet 2/1
    ip address 20.20.1.1 255.255.255.0
!
```

Templates for these CLIs would be:

```
!
interface FastEthernet %FastEthernet 0%
    ip address 10.10.1.1 255.255.255.0
!
interface FastEthernet %FastEthernet 1%
    ip address 20.20.1.1 255.255.255.0
!
```

Example 2 (Voice card with two ports plugged in slot 3):

```
!
voice-port 3/0/0
    description 4082224444
!
voice-port 3/0/0
    description 4082225555
!
```

Templates for these CLIs would be:

```
!
voice-port  %voice-port 0%
    description 4082224444
!
voice-port %voice-port 1%
    description 4082225555
!
```

The main device template does not include links to the subdevice templates. The subdevice templates are appended to the main device template. The line card numbers are a parameter in the subdevice templates.

All the CLI commands which reference a line card interface are specified in the subdevice template for that line card. This implies that any command in the global configuration mode, or otherwise, that refers to a particular line card interface is in the template for that subdevice (line card) and not in the main device template.

Only the CLI commands in the global configuration mode, and not pertaining to the any specific interface, are specified in the main device template.

The port number and channel number are not template parameters since these are fixed for a given line card. The network administrator can configure specific channels on the interfaces by explicitly specifying the channels in the subdevice templates.

For example:

interface Serial %Serial 0%:0

Sample Templates for Modular Router

The names of the attributes for slot, slot-unit, line card type and so forth, are used for demonstration purposes.

Main Device Template

```
!
version 12.2
no parser cache
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname 2600
!
```

```
logging rate-limit console 10 except errors
!
memory-size iomem 25
ip subnet-zero
!
!
!
no ip dhcp-client network-discovery
lcp max-session-starts 0
!
ip classless
no ip http server
!
call rsvp-sync
!
no mgcp timer receive-rtcp
!
mgcp profile default
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
line vty 5 15
 login
!
```

FastEthernet Template

```
Interface FastEthernet %FastEthernet 0%

ip address 10.0.0.1 255.0.0.0
shutdown
speed auto
```

Voice-port Template

```
voice-port  %voice-port 0%
playout-delay mode adaptive
!
voice-port %voice-port 1%
!
dial-peer voice 10 pots
destination-pattern 200
port %voice-port 0%
forward-digits all


voice-port  %voice-port 0%
!
dial-peer voice 20 pots
destination-pattern 100
port %voice-port 0%
!
voice-port  %voice-port 1%
```

Modular Router Events

Modular router events are published to the event bus and are accessible to applications connected to the bus. The IOS device publishes the system hardware configuration in the *cisco.cns.config.device-details* event after hardware discovery. The Cisco Configuration Engine is configured to listen for this event, retrieve it, and extract the hardware configuration of the device.

Following is the DTD of the *cisco.cns.config.device-details* event that the Cisco IOS device sends:

```
    <!ELEMENT device-details (config-id, connect-interface?, card-info*>
    <!ELEMENT config-id (#PCDATA)>
    <!ELEMENT connect-interface (#PCDATA)>
    <!ELEMENT card-info (card-info+)>
    <!ELEMENT card-info
(card-type,card-desc?,slot,daughter?,serial-number,part-number,hw-version?,board-revision?
,ports?,controller?,rma-number?,test-history?,eeprom-version?,eeprom-data?,interface?,cont
roller?,voice-port?)>
    <!ELEMENT card-type (#PCDATA)>
    <!ELEMENT card-desc (#PCDATA)>
    <!ELEMENT slot (#PCDATA)>
    <!ELEMENT daughter (#PCDATA)>
    <!ELEMENT serial-number (#PCDATA)>
    <!ELEMENT part-number (#PCDATA)>
    <!ELEMENT hw-version (#PCDATA)>
    <!ELEMENT board-revision (#PCDATA)>
    <!ELEMENT ports (#PCDATA)>
    <!ELEMENT controller (#PCDATA)>
    <!ELEMENT rma-number (#PCDATA)>
    <!ELEMENT test-history (#PCDATA)>
    <!ELEMENT eeprom-version (#PCDATA)>
    <!ELEMENT eeprom-data (#PCDATA)>
    <!ELEMENT interface (#PCDATA)>
    <!ELEMENT controller (#PCDATA)>
    <!ELEMENT voice-port (#PCDATA)>
```

Dynamic Templates

There might be times when the actual contents of a template needs to be dynamically generated. To do this, you would use the **#call** mechanism. This executes a JavaScript program whose output becomes part of the template. The program is re-executed each time a device asks for the template.

For example, you might want to distribute the load across the various event gateway processes without permanently assigning a device to a particular event gateway. This is useful because of the limit of 500 devices per event gateway daemon instance.

Let us take the following template as an example:

```
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname DemoRouter
#call /opt/CSCOcnsie/Templates/event_setup.js
```

Here is an example of an *event_setup.js* that one might use:

```
/*
 * An instance of Event Gateway resides on every odd port from 11011 to 11031.
 * This will choose a random one in this range so that devices are spread out
 * evenly among the various ports. Adjust the IP address in the println
 * statement to be the address of the IE2100 itself.
 */
var port = Math.floor(Math.random() * 11) * 2 + 11011;
println("cns event 10.1.6.131 " + port.toString());
```

The result of this combination would be a template that appears as follows:

```
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname DemoRouter
cns event 10.1.6.131 11017
```

The last line is programmatically determined and recalculated every time the template is requested by the device. So the next time a device requests this template, the last line might be:

```
cns event 10.1.6.131 11023
```

Simple modifications to *event_setup.js* could even be used to distribute devices across multiple Cisco 2116 devices (by dynamically generating the IP address). It could also be used to affect any part of the device configuration—be it DNS servers or routing tables. Anything that is printed out by the JavaScript program becomes a dynamic part of the template.

Control Structures

The configuration template can include simple control structures such as, *if*, *else* and *elseif*. By using these control structures, the user can include or exclude a block of CLI commands based on a parameter stored in the directory.

The syntax for these # preprocessing control structures is as follows:

Syntax Description

#if <*URL*> = *constant*

cli-command(s)

#elseif <*URL*> = *constant*

cli-command(s)

#else

cli-command(s)

#endif

Where *constant* is an integer, boolean or a string in single quotes and the <*URL*> is a URL pointing to an attribute in the Directory or Database.

Note Nested **#if** and **#elseif** is NOT supported.

Usage Guidelines

The configuration template can include **#define** entries to define short names for long URLs.

The syntax for the **#define** preprocessing command is as follows:

#define *definition-name <URL>* | *constant*

where *<URL>* is a reference to an attribute in the directory.

The configuration template can contain another # preprocessing command **#include,** which allows the inclusion of other configuration templates or the results of an ASP page.

The syntax for the # preprocessing command is as follows:

#include *<URL>* | '*<Filename>*' | *<Filename>*

Whenever an **#include** directive is encountered, it is replaced by the content of the file.

The following configuration template sample includes either IP sub-template or ISDN sub-template based on the value of the parameter protocol in the directory or database.

Examples

```
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname ${LDAP://this:attrName=IOShostname}
#if ${LDAP://this:attrName=IOSIPprotocol} = true then
    #include ${LDAP://this:attrName=IPsubTemplate}
#else
    #include ${LDAP://this:attrName=ISDNsubTemplate}
#endif
```

The parameter, `${LDAP://this:attrName=IPsubTemplate}` contains the location of the file.

Managing Templates

To access Template management tasks, log into the system (see "Logging In and Out" section on page 2-1). Then, from the Home page, click the **Tools** tab. The Tools page appears.

From the Tools page, click **Template Mgr**. The Template Manager page appears showing:

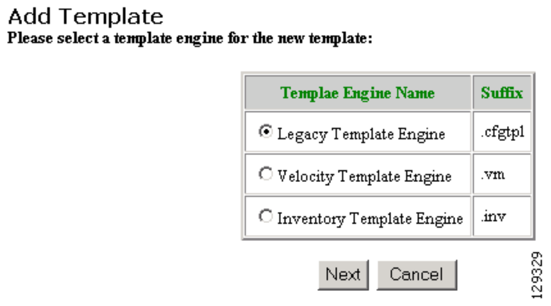
- Add Template
- Edit Template
- Delete Template
- Import Template

Adding a Template

Step 1 From the Template Manager page, click **Add Template**.

The Template Engine page appears (see Figure 12-1).

Figure 12-1 Template Engine



Add Template

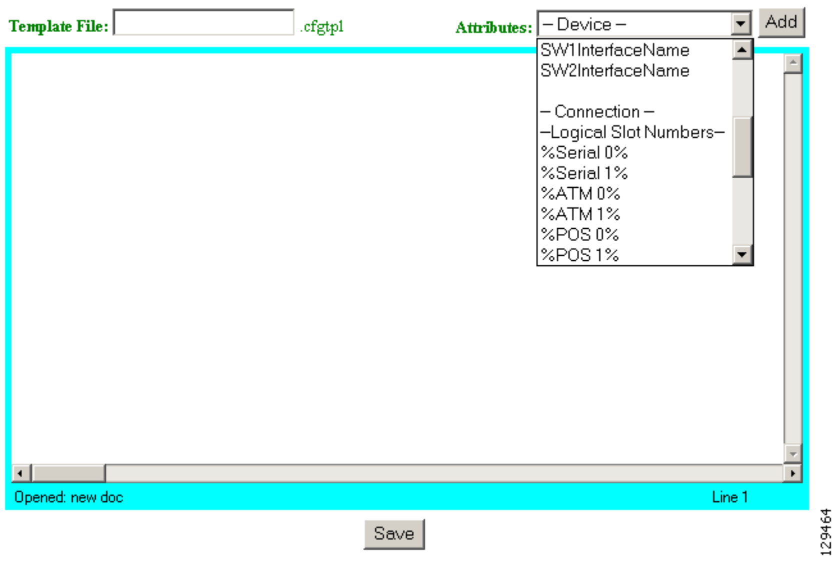
Please select a template engine for the new template:

Templae Engine Name	Suffix
● Legacy Template Engine	.cfgtpl
○ Velocity Template Engine	.vm
○ Inventory Template Engine	.inv

Select the Template Engine for the new template, then click **Next**.

A blank template page appears (see Figure 12-2).

Figure 12-2 Blank Template Page



Template File: .cfgtpl

Attributes: – Device –

SW1InterfaceName
SW2InterfaceName

– Connection –
–Logical Slot Numbers–
%Serial 0%
%Serial 1%
%ATM 0%
%ATM 1%
%POS 0%
%POS 1%

Opened: new doc Line 1

Step 2 Enter the filename for this template in the **Template File** field.

Table 12-1 shows valid values for these fields.

Table 12-1 Valid Values for Add Template

Attribute	Description	Valid Values
Template File	Filename of template	a-z A-Z 0-9 -(hyphen) _ (under-score) . (period)
Attributes	Available attributes	From drop-down list

Step 3 To choose the attributes you want to be included in this template, use the **Attributes** menu.

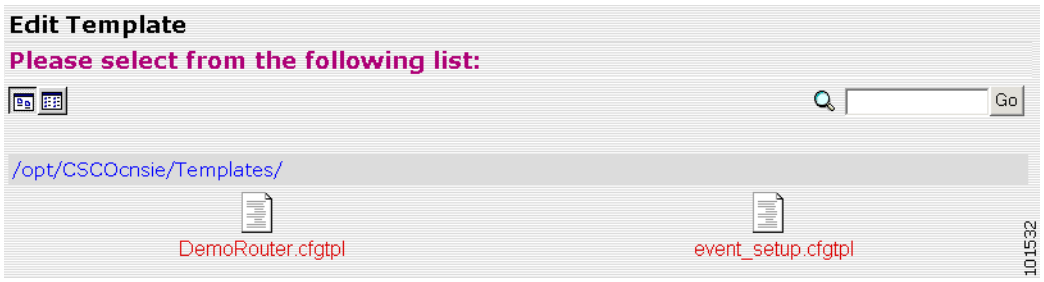
Step 4 To save your entries, click **Save**.

Editing a Template

Step 1 From the Template Manager page, click **Edit Template**.

The Edit Template list appears (see Figure 12-3).

Figure 12-3 Edit Template List


Edit Template
Please select from the following list:
/opt/CSCOcnsie/Templates/
DemoRouter.cfgtpl
event_setup.cfgtpl

Step 2 Click on the icon for the template file you want to edit.

The template file appears.

Step 3 To edit parameters (attribute information):

a. From the template file page, click **Edit AttributeInfo**.

b. Edit the desired parameter fields.

c. To clear your entries, click **Reset**.

d. To save your changes, click **Save**.

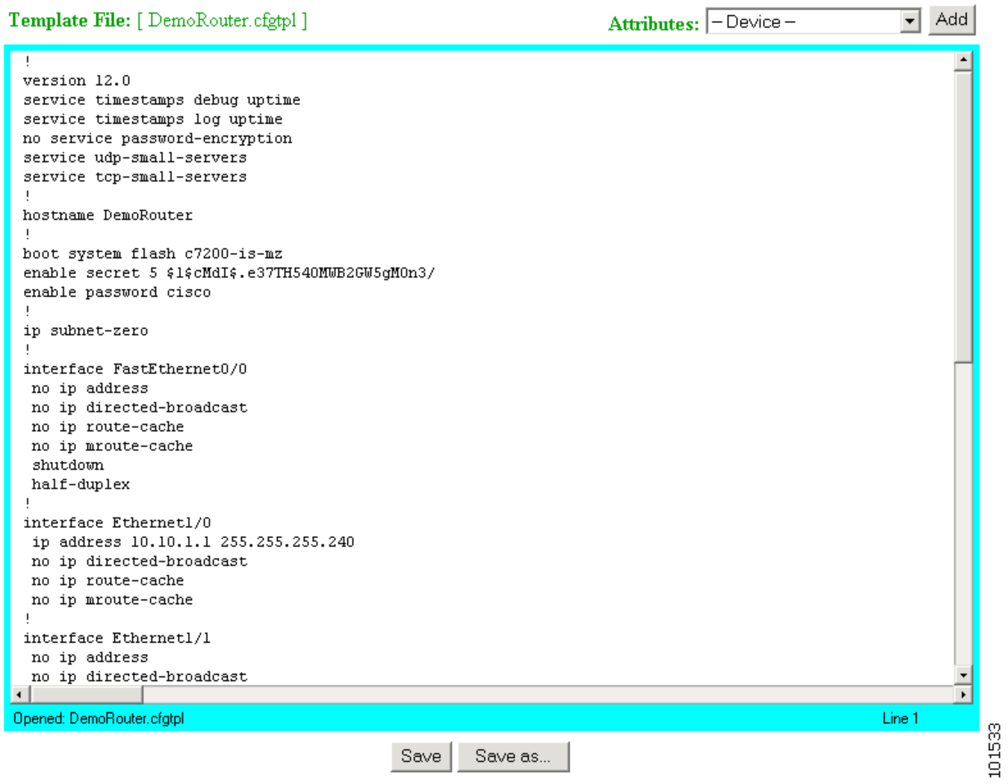
Step 4 To save and apply, **Save and Apply**.

Step 5 To edit template content:

a. To edit the content of a template, from the template file page, click **Edit Content**.

The template content appears (see Figure 12-4).

Figure 12-4 Template Content


Template File: [DemoRouter.cfgtpl]
Attributes:
– Device –
Add
Opened: DemoRouter.cfgtpl
Line 1
Save
Save as...
101533

```
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname DemoRouter
!
boot system flash c7200-is-mz
enable secret 5 $1$cMdI$.e37TH540MWB2GW5gMOn3/
enable password cisco
!
ip subnet-zero
!
interface FastEthernet0/0
 no ip address
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
 shutdown
 half-duplex
!
interface Ethernet1/0
 ip address 10.10.1.1 255.255.255.240
 no ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
!
interface Ethernet1/1
 no ip address
 no ip directed-broadcast
```

b. Edit the content by adding or deleting attributes.

c. To save your edits, click **Save**.

d. To save as a new template, click **Save as**.

Deleting a Template

Step 1 From the Template Manager page, click **Delete Template**.

The template file list appears.

Step 2 Select the template you want to delete.

Step 3 Delete the desired template file.

Importing a Template

Step 1 From the Template Manager page, click **Import Template**.

Step 2 In the dialog box that appears, enter the name of the template file in the **Filename** field, if known, or browse your directory tree to choose the filename you desire.

Step 3 To clear the field, click **Reset**.

Step 4 To upload the template file, click **Upload**.