CHAPTER 12

テンプレート

テンプレートを作成する場合、コンテキストに合わせて置き換わる変数を指定できます。これらの変数の多くは、テンプレート エディタ（図 12-4 を参照）のドロップダウン メニューから使用できます。また、テンプレート エディタを使用せずにこれらのファイルをオフラインで作成しても、これらの変数を使用できます。

テンプレート ファイルの基本的な形式は、デバイスにダウンロードする設定の単純なテキストです（「サンプル テンプレート」（P.12-1）を参照）。しかし、次の形式で変数の置換を配置できます（この例では、変数の名前として *iosipaddress* を使用します）。

```
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}
```

テンプレートのセグメントを作成し、他のテンプレートに組み込むことができます。たとえば、複数のデバイスで使用されるイーサネット設定があるとします。各デバイス テンプレートに次の行を含めます。

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

この方法で、イーサネット設定に関するすべての管理を 1 つのファイルに統合できます。

注意 組み込むテンプレート ファイルの循環は許可されません。

サンプル テンプレート

次に、システム上に事前ロードされる DemoRouter の設定テンプレート（*DemoRouter.cfgtpl*）のサンプルを示します。

```
!
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
```

設定コントロール テンプレート

新しいイメージを使用してデバイスを再起動するには、特定のデバイス上でのイメージ アクティベーションのために必要な CLI コマンドを含む設定コントロール テンプレートが必要です。

たとえば、Cisco 3600 シリーズ ルータを *3600.image* という名前のイメージでデバイス コンソールから再起動する場合、次の CLI コマンドを発行します。

no boot system
boot system flash:3600.image

イメージ アクティベーションのための設定コントロール テンプレートの内容には、通常デバイス コンソールから入力して新しいイメージをデバイス上でアクティブにする、CLI コマンドが含まれている必要があります。

ダイナミック フロー コントロール テンプレート

イメージ エージェントで収集されたインベントリ情報は、ダイナミック フロー コントロール テンプレートによって外部ユーザに対して使用可能になります。これにより、インベントリ情報に基づいて、設定フローとイメージの配信ジョブを制御できるテンプレートを記述できます。

インベントリの操作

ダイナミック フロー コントロール テンプレートからデバイスのインベントリにアクセスするため、ユーザに対して次の操作が公開されています。

関数	**$!{invObj.getDram()}**
返される型	int（バイト単位）。
説明	Dram = メイン メモリ サイズ + IO メモリ サイズ。 DRAM のサイズを返します。

関数	**$!{invObj.getVersionString()}**
返される型	文字列型。
説明	デバイス インベントリからの現在実行されているイメージのバージョン文字列を返します。

関数	**$!{invObj.getImageFile()}**
返される型	文字列型。
説明	現在実行されているイメージのファイル名を返します。

関数	$!{invObj.getImageMD5()}
返される型	文字列型。
説明	デバイス インベントリ内で提供されているように MD5 を返します。

関数	$!{invObj.getStartedAt()}
返される型	文字列型。
説明	デバイスの起動時の時間文字列を返します。

関数	$!{invObj.getPlatformName()}
返される型	文字列型。
説明	プラットフォーム名を返します。

関数	$!{invObj.getFlash()}
返される型	int（バイト単位）。
説明	フラッシュのサイズを返します。

関数	$!{invObj.getFileSysSize("bootflash")}
返される型	int（バイト単位）。
説明	ブートフラッシュのサイズを返します。

関数	$!{invObj.getFileSysFreespace("bootflash")}
返される型	int（バイト単位）。
説明	ブートフラッシュの空き領域の量を返します。

関数	$!{invObj.getFileSysSize("nvram")}
返される型	int（バイト単位）。
説明	NVRAM のサイズを返します。

関数	$!{invObj.getFileSysFreespace("nvram")}
返される型	int（バイト単位）。
説明	NVRAM の空き領域の量を返します。

関数	$!{invObj.getFileSysSize("disk0")}
返される型	int（バイト単位）。
説明	disk0 のサイズを返します。

関数	$!{invObj.getFileSysFreespace("disk0")}
返される型	int（バイト単位）。
説明	disk0 の空き領域の量を返します。

関数	$!{invObj.getFileSysSize("slot0")}
返される型	int（バイト単位）。
説明	slot0 のサイズを返します。

関数	$!{invObj.getFileSysFreespace("slot0")}
返される型	int（バイト単位）。
説明	slot0 の空き領域の量を返します。

関数	$!{invObj.getFileSysSize("slot1")}
返される型	int（バイト単位）。
説明	slot1 のサイズを返します。

関数	$!{invObj.getFileSysFreespace("slot1")}
返される型	int（バイト単位）。
説明	slot1 の空き領域の量を返します。

その他の操作

ダイナミック フロー コントロール テンプレートから上記の条件に基づいてアクションを実行するため、ユーザに対して次の操作が公開されています。

関数	$!{cnsceObj.distribute()}
パラメータ	なし。
説明	イメージの配信を実行します。事前設定されたイメージが使用されます。

関数	$!{cnsceObj.activate("persist" | "nv_overwrite")}
パラメータ	設定アクションを次のように設定します。 • “persist”：設定を適用して NVRAM に保存します。 • “nv_overwrite”：NVRAM 設定を上書きします。
説明	イメージのアクティベーションを実行します。事前設定されたイメージが使用されます。

関数	$!{cnsceObj.updateConfig(true | false, "write" | “persist” | “nv_overwrite”)}
パラメータ	最初のパラメータに、次のように構文チェックを設定します。 • true：構文チェックをオンにします。 • false：構文チェックをオフにします。 2 番目のパラメータに、次のように設定アクションを設定します。 • “write”：実行中の設定に適用します。 • “persist”：設定を適用して NVRAM に保存します。 • “nv_overwrite”：NVRAM 設定を上書きします。
説明	設定の更新を実行します。事前設定されたテンプレートが使用されます。

注意

invObj.getDram() 操作では次の情報が返されます。

Dram = メイン メモリ サイズ + IO メモリ サイズ

例

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

上記の例に示すように、DRAM サイズに応じてジョブのフローをカスタマイズできます。

上記のインベントリ テンプレートを使用したカスタム ジョブが発行されると、デバイスはそのデバイスのインベントリに対して照会され、DRAM サイズに応じて、イメージのアップグレードを実施するかどうかが判断されます。上記の例のインベントリ テンプレートが評価された結果、デバイスの DRAM サイズが 6100 バイトを超えている場合、イメージの配信とイメージのアクティベーションが実施されます。

サンプル 1

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

サンプル 2

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

サンプル 3

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

モジュラ ルータ用のテンプレート

デバイスのテンプレート メカニズムは、モジュラ ルータをサポートするように拡張されています。モジュラ ルータのシャーシには、モジュールを実装するためのスロットが備わっています。このシャーシ内の使用可能な任意のスロットに、任意のモジュールを実装できます。2 つのイーサネット 2 WAN カード スロット モジュールなどの一部のモジュールは、インターフェイス カードまたはラインカードを実装するためのサブ スロットが同様にあります。デバイス管理は、ラインカードに相当するサブデバイスをサポートするように拡張されました。

メイン デバイスまたはサブデバイスを表す目的で同じ構造を持たせるために、ラインカード番号、ラインカード タイプ、およびサブデバイスに相当する追加のアトリビュートが、ディレクトリ サーバ内の既存のデバイス オブジェクト構造に追加されました。

現在、カード タイプは、ネットワーク モジュールの製品コードにマッピングされる文字列です。カード内の EPROM データには、製品コードでなく製品番号だけが格納されているため、製品番号は製品コードにマッピングされます。ユーザは製品番号を使用し、コンフィギュレーション サーバは製品番号を製品コードにマッピングします。

メイン デバイスの場合、ラインカード番号フィールドおよびラインカード タイプ フィールドは使用されないため、ヌル値に設定されます。サブデバイス内のサブデバイス フィールド（ラインカードに相当する）は、ヌル値に設定されます。

新しいインターフェイス変数のサポートが追加されました。これらの変数はテンプレートに含められ、テンプレート内のインターフェイス番号を使用してパラメータ化されます。これらはアトリビュートではありません。これらは特殊形式の変数であり、デバイスからのインターフェイス情報に基づいてコンフィギュレーション サーバによって置換されます。これらの変数はモジュール上のインターフェイスの相対的な位置を指定するだけであり、実際のスロット番号、シェルフ ID、またはポート番号に置換されます。インターフェイス変数はパーセント記号（%）文字で囲まれ、該当する場合はタイプと、相対的な位置を指定します。コンフィギュレーション サーバはこれらの変数をインターフェイス番号に置換します。この場合も、インターフェイス タイプは次の構文を使用して CLI で指定する必要があります。

インターフェイス変数 = %[InterfaceType] RelativePosition%

例：

%FastEthernet 0%：FastEthernet インターフェイス用

%Serial 0%：シリアル インターフェイス

%T1 0%：コントローラ T1

%E1 0%：コントローラ E1

%voice-port 0%：ボイス ポート

例 1：

スロット 2 に取り付けられた 2 つの FastEthernet ポートがあるネットワーク モジュールは、設定 CLI 内で次のように FastEthernet 2/0 および FastEthernet 2/1 として参照され、テンプレート内で FastEthernet %FastEthernet 0% および FastEthernet %FastEthernet 1% として参照されます。

```
!
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
!
```

これらの CLI のテンプレートは次のようになります。

```
!
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
!
```

例 2（スロット 3 に取り付けられた 2 つのポートがあるボイス カード）：

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

これらの CLI のテンプレートは次のようになります。

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

メイン デバイス テンプレートにはサブデバイス テンプレートへのリンクは含まれません。サブデバイス テンプレートは、メイン デバイス テンプレートに付加されます。ラインカード番号は、サブデバイス テンプレート内のパラメータです。

ラインカード インターフェイスを参照する、すべての CLI コマンドは、そのラインカードのためのサブデバイス テンプレート内で指定されます。これは、グローバル コンフィギュレーション モードであるかどうかにかかわらず、特定のラインカード インターフェイスを参照する任意のコマンドは、そのサブデバイス（ラインカード）用のテンプレート内にあり、メイン デバイス テンプレート内にはないことを意味します。

グローバル コンフィギュレーション モード内の CLI コマンドで、任意の特定のインターフェイスに関係しないコマンドだけが、メイン デバイス テンプレート内に指定されます。

ポート番号およびチャネル番号は、特定のラインカードに対して固定されたもののため、テンプレート パラメータではありません。ネットワーク管理者は、サブデバイス テンプレート内にチャネルを明示的に指定することで、インターフェイス上に特定のチャネルを設定できます。

例：

interface Serial %Serial 0%:0

モジュラ ルータ用のサンプル テンプレート

スロット、スロットユニット、ラインカード タイプなどのためのアトリビュート名は、デモの目的で使用されます。

メイン デバイス テンプレート

```
!
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 テンプレート

```
Interface FastEthernet %FastEthernet 0%

ip address 10.0.0.1 255.0.0.0
shutdown
speed auto
```

ボイス ポート テンプレート

```
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%
```

モジュラ ルータ イベント

モジュラ ルータ イベントがイベント バスに発行され、そのバスに接続するアプリケーションからアクセス可能になります。IOS デバイスは、ハードウェアの検出後にシステム ハードウェア設定を *cisco.cns.config.device-details* イベント内で発行します。Cisco Configuration Engine はこのイベントをリッスンし、イベントを取得してデバイスのハードウェア設定を抽出するように設定されます。

次に、Cisco IOS デバイスが送信する *cisco.cns.config.device-details* イベントの DTD を示します。

```
    <!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)>
```

動的なテンプレート

場合によっては、テンプレートの実際の内容は動的に生成される必要があります。これを行うには、**#call** メカニズムを使用できます。このメカニズムを使用することで、JavaScript プログラムを実行し、その出力をテンプレートの一部にすることができます。このプログラムは、デバイスがテンプレートを要求するたびに再実行されます。

たとえば、特定のイベント ゲートウェイに対してデバイスを固定的に割り当てずに、さまざまなイベント ゲートウェイ処理に負荷を分散したい場合があります。これを行うと、イベント ゲートウェイ デーモン インスタンスごとに 500 台のデバイス制限があるために有効です。

次のテンプレートを例として検討します。

```
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
```

このような場合に使用する例として、*event_setup.js* を挙げます。

```
/*
 * 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());
```

この組み合わせの結果、次のようなテンプレートが作成されます。

```
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
```

最後の行はプログラム上で決定され、デバイスがテンプレートを要求するたびに再計算されます。そのため、次回デバイスがこのテンプレートを要求したときは、最後の行は次のようになることがあります。

```
cns event 10.1.6.131 11023
```

event_setup.js を変更するだけで、（IP アドレスを動的に生成することで）デバイスを複数のホスト デバイス間に分散するために使用することもできます。また、DNS サーバやルーティング テーブルなど、デバイス設定の任意の部分に反映するように使用できます。JavaScript プログラムで出力される部分は、テンプレートの中の動的に変化する部分として使用できます。

コントロール構造

設定テンプレートには、*if*、*else*、および *elseif* などのコントロール構造を含めることができます。これらのコントロール構造を使用することで、ユーザはディレクトリに保存されたパラメータに基づいて、CLI コマンドのブロックをインクルードする、または除外することができます。

前処理コントロール構造の構文は次のとおりです。

シンタックスの説明

#if <*URL*> = *constant*

　CLI コマンド

#elseif <*URL*> = *constant*

　CLI コマンド

#else

　CLI コマンド

#endif

constant には、単一引用符で囲んだ整数、ブール値、または文字列を指定し、<*URL*> にはディレクトリまたはデータベース内のアトリビュートを指す URL を指定します。

（注） ネストされた **#if** および **#elseif** はサポートされません。

使用上のガイドライン

設定テンプレートには、長い URL に対して短い名前を定義するために **#define** エントリをインクルードできます。

#define 前処理コマンドの構文は次のとおりです。

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

<*URL*> に、ディレクトリ内のアトリビュートへの参照を指定します。

設定テンプレートには、別の # 前処理コマンドである **#include** を含めることができ、これを使用して他の設定テンプレートまたは ASP ページの結果をインクルードできます。

前処理コマンドの構文は次のとおりです。

#include <*URL*> | ‘<*Filename*>*”* | <*Filename*>

#include ディレクティブがあると、そのディレクティブは常にそのファイルの内容で置き換えられます。

次の設定テンプレート サンプルには、ディレクトリまたはデータベース内のパラメータ プロトコルに基づき、IP サブテンプレートまたは ISDN サブテンプレートがインクルードされます。

例

```
!
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
```

`${LDAP://this:attrName=IPsubTemplate}` パラメータにファイルの場所が含まれます。

テンプレートの管理

テンプレート管理タスクにアクセスするには、システムにログインします（「ログイン」（P.2-1）を参照）。次に、[Home] ページから、[Tools] タブをクリックします。[Tools] ページが表示されます。

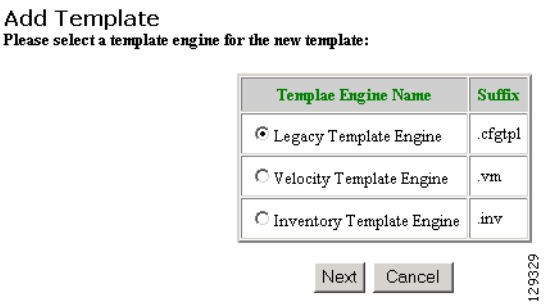
[Tools] ページから、[Template Mgr] をクリックします。[Template Manager] ページに次の項目が表示されます。

- [Add Template]
- [Edit Template]
- [Delete Template]
- [Import Template]
- [Export Template]
- [Import Local Template]

テンプレートの追加

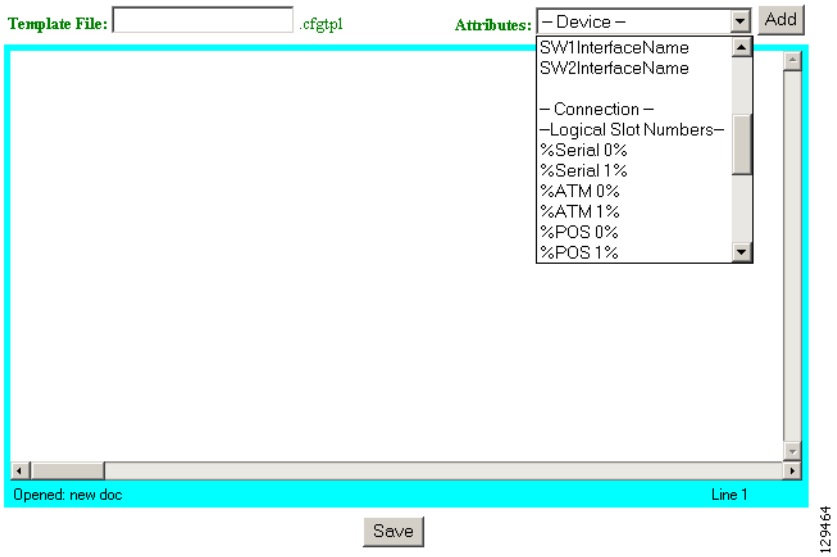
ステップ 1　[Template Manager] ページから、[Add Template] をクリックします。
[Template Engine] ページが表示されます（図 12-1 を参照）。

図 12-1　[Template Engine]



新しいテンプレートのテンプレート エンジンを選択し、[Next] をクリックします。
空のテンプレート ページが表示されます（図 12-2 を参照）。

図 12-2　空のテンプレート ページ



ステップ 2　このテンプレートのファイル名を [Template File] フィールドに入力します。
表 12-1 に、これらのフィールドに使用できる値を示します。

表 12-1　[Add Template] の有効な値

アトリビュート	説明	有効な値
Template File	テンプレートのファイル名	a ～ z A ～ Z 0 ～ 9 -（ハイフン） _（アンダースコア） .（ピリオド）
Attributes	使用可能なアトリビュート	ドロップダウン リストから選択

ステップ 3　このテンプレートに含めるアトリビュートを選択するには、[Attributes] メニューを使用します。

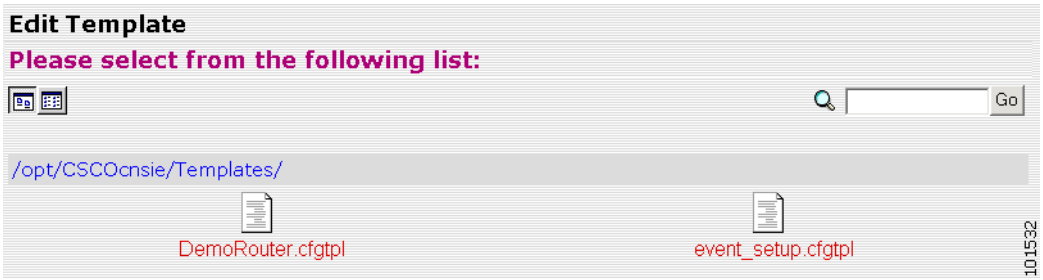
ステップ 4　エントリを保存するには、[Save] をクリックします。

テンプレートの編集

ステップ 1　[Template Manager] ページから、[Edit Template] をクリックします。

[Edit Template] リストが表示されます（図 12-3 を参照）。

図 12-3　[Edit Template] リスト



ステップ 2　編集するテンプレート ファイルのアイコンをクリックします。

テンプレート ファイルが表示されます。

ステップ 3　パラメータ（アトリビュート情報）を編集するには、次の操作を実行します。

a.　テンプレート ファイル ページから、[Edit AttributeInfo] をクリックします。

b.　目的のパラメータ フィールドを編集します。

c.　エントリをクリアするには、[Reset] をクリックします。

d.　変更を保存するには、[Save] をクリックします。

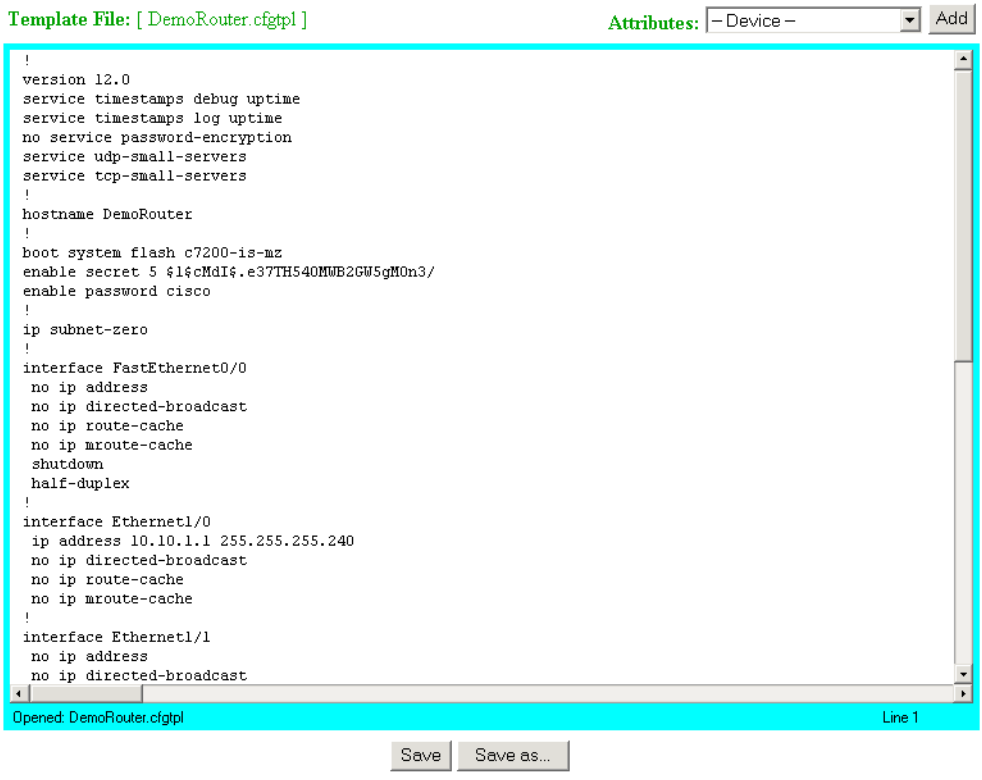
ステップ 4　保存して適用するには、[Save and Apply] をクリックします。

ステップ 5　テンプレートの内容を編集するには、次の操作を実行します。

a.　テンプレートの内容を編集するには、テンプレート ファイル ページから、[Edit Content] をクリックします。

テンプレートの内容が表示されます（図 12-4 を参照）。

図 12-4　テンプレートの内容



b. アトリビュートを追加または削除して、この内容を編集します。

c. 編集内容を保存するには、[Save] をクリックします。

d. 新しいテンプレートとして保存するには、[Save as] をクリックします。

テンプレートの削除

ステップ 1　[Template Manager] ページから、[Delete Template] をクリックします。
テンプレート ファイル リストが表示されます。

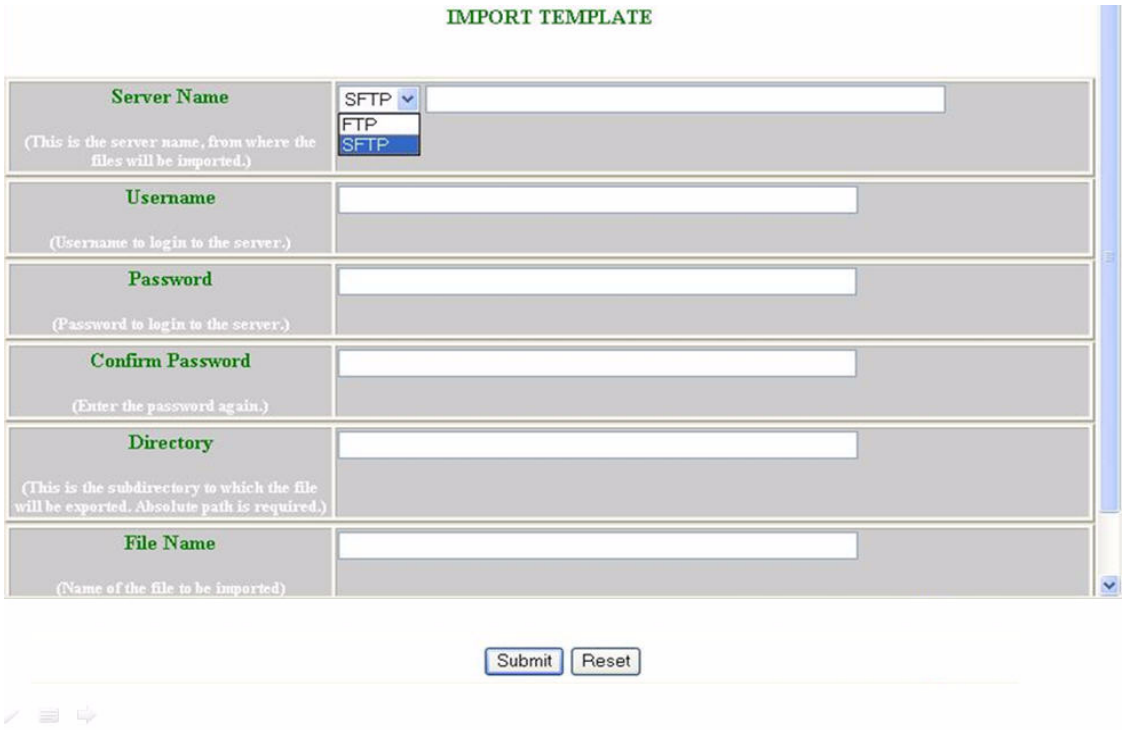
ステップ 2　削除するテンプレートを選択します。

ステップ 3　目的のテンプレート ファイルを削除します。

テンプレートのインポート

ステップ 1　[Template Manager] ページから、[Import Template] をクリックします。

図 12-5　　テンプレートのインポート



ステップ 2　表示されるダイアログボックスで、ファイルのエクスポート先のサーバ名の FTP または SFTP パスを [FTP/SFTP] フィールドに入力します。

ステップ 3　ユーザ名を [Username] フィールドに入力します。

ステップ 4　パスワードを [Password] フィールドに入力します。

ステップ 5　新しいパスワードを [Confirm Password] フィールドに再入力します。

ステップ 6　ファイルのエクスポート先のサブディレクトリ パスを [Director] フィールドに入力します。

ステップ 7　テンプレート ファイルのファイル名がわかっている場合、その名前を [Filename] フィールドに入力します。わからない場合、ディレクトリ ツリーを参照して目的のファイル名を選択します。

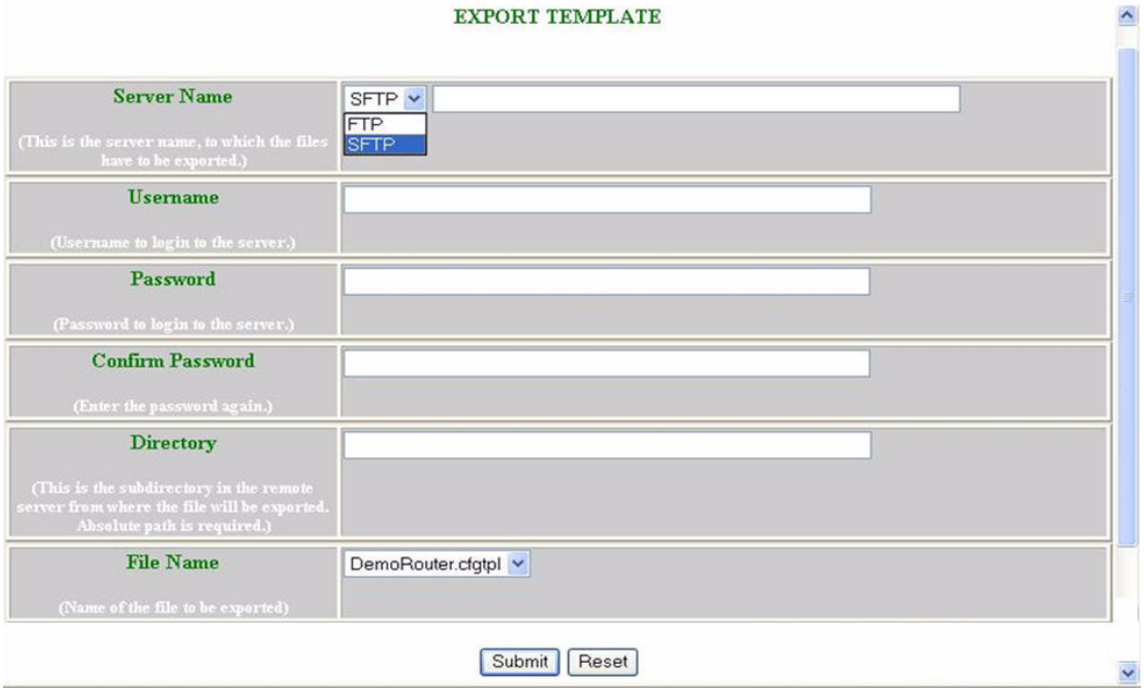
ステップ 8　このフィールドをクリアするには、[Reset] をクリックします。

ステップ 9　テンプレート ファイルをインポートするには、[Submit] をクリックします。

テンプレートのエクスポート

ステップ 1　[Template Manager] ページから、[Export Template] をクリックします。

図 12-6　　テンプレートのエクスポート



ステップ 2　表示されるダイアログボックスで、ファイルのエクスポート先のサーバ名の FTP または SFTP パスを [FTP/SFTP] フィールドに入力します。

ステップ 3　ユーザ名を [Username] フィールドに入力します。

ステップ 4　パスワードを [Password] フィールドに入力します。

ステップ 5　新しいパスワードを [Confirm Password] フィールドに再入力します。

ステップ 6　ファイルのエクスポート先のサブディレクトリ パスを [Director] フィールドに入力します。

ステップ 7　テンプレート ファイルのファイル名がわかっている場合、その名前を [Filename] フィールドに入力します。わからない場合、ディレクトリ ツリーを参照して目的のファイル名を選択します。

ステップ 8　このフィールドをクリアするには、[Reset] をクリックします。

ステップ 9　テンプレート ファイルをインポートするには、[Submit] をクリックします。

ローカル テンプレートのインポート

ステップ 1　[Template Manager] ページから、[Import Local Template] をクリックします。

図 12-7　[Import Local Template]

ステップ 2　テンプレート ファイルのファイル名がわかっている場合、その名前を [Filename] フィールドに入力します。わからない場合、ディレクトリ ツリーを参照して目的のファイル名を選択します。

ステップ 3　テンプレート ファイルをインポートするには、[Upload] をクリックします。