



Case Studies on CiscoWorks Resource Manager Essentials Compliance Management

White Paper

Contents

Feature Overview	3
Case Studies	3
Case Study 1	3
Case Study 2	6
Use Cases	8
Additional Use Cases	9
Case 1: Compliance check on community strings	9
Case 2: Identifying devices configured with incorrect SNMP trap strings	10
Case 3: Checking for devices not configured with TACACS	10
Case 4: Replacing helper addresses	11
Case 5: Adding a default configuration set on shutdown interfaces using the basic template	15
Case 6: Compliance check for routing protocol configurations	16
Case 7: Search and replace	18
Case 8: Using an ordered set to check for ACL compliance	18
Case 9: Deploy using an XML parameter file	19
FAQ	21
Conclusion	21

Feature Overview

The baseline template is a feature in CiscoWorks Resource Manager Essentials (RME) that allows the network administrator to identify and select a set of commands for checking network compliance. These templates can be customized to meet organizational policy requirements and run on a group of devices in the network.

A baseline template consists of a set of commands that the user has identified from the defined policies. The user can specify the commands that should be checked across devices and can provide parameterized placeholders for those values that are specific to particular devices.

The feature helps you to achieve the following:

- Check for device configuration compliance
- Perform periodic checks on devices that are getting added on to your network
- Perform periodic reporting on noncompliant devices and help network administrators deploy configurations that make the devices compliant
- Export template commands to XML format
- Help import external configuration commands in XML format

Case Studies

Case Study 1

CiscoWorks provides various features to assist with configuration compliance. One feature was used for overcoming the challenges that one of our financial customers had. The following were the business challenges faced by the organization under discussion:

- Rapid network expansion
- Internal staff restructuring
- Limited risk management automation capabilities
- Inefficient response to regulatory compliance and inventory management requirements

In an environment of rapid network expansion coupled with operating expenditure constraints, there is considerable pressure on IT in managing the network in an efficient and productive way. Under a climate of operating expenses (OpEx) reduction it is difficult to manually implement network configuration standards based on Cisco® leading practices.

Solution

The baseline template feature in Resource Manager Essentials is an end-to-end workflow change and configuration management solution that allows enterprises to automate change configuration tasks, validates network configurations, and helps to maintain industry and regulatory compliance.

The baseline template feature of RME has enabled this customer to administer 3000 production network changes in less than a 2-day person effort.

There were two major aspects of configuration compliance:

1. Ensuring that device configurations are as per company policy
 - Configuration compliance is an iterative process. There are a set of rules that the company has defined based on the configuration best practices given by Cisco.
 - The process for implementing device configuration compliance starts with configuration templates being created with the desired set of configurations.
 - Compliance verification is performed with the most critical items first, and the results achieved at each stage make it easier to progress.
2. Auditing the configurations to validate adherence to standard company policies

One important challenge in auditing the configurations arises with the nuances of different Cisco IOS® Software versions. For example the default values for features in Cisco IOS Software can change between versions. To deal with these Cisco IOS variations and achieve configuration compliance, CiscoWorks RME features were used to work with Cisco IOS Software and with the company's change process.

This was done with two features of CiscoWorks RME:

- NetConfig jobs and user-defined tasks
- Compliance management using baseline templates

The first step is to create templates and NetConfig jobs that will deploy the required configuration to devices. Then a baseline template is created to verify what devices need to have what configuration changed. The most effective manner to deal with them is to deploy the configuration to a group of existing devices one at a time, and then validate the status using the baseline templates.

Because baseline templates can potentially generate thousands of exceptions, it is best to break these up into manageable chunks. It is a best practice to understand the impact of configuration changes before deploying them on the devices.

1. Initially the basic NetConfig user-defined tasks were created to make sure the devices have been configured correctly for manageability. These include configurations pertaining to syslogs, access control lists (ACLs), Simple Network Management Protocol (SNMP), and other basic switch and router configurations.
2. The next step was to check and validate whether these configurations are present in the network devices using the baseline template feature.

One requirement of the customer was to have version control of security standards, which requires a mapping of releases and changes to versions of baseline templates. It was necessary to rename the templates after changes were made. CiscoWorks RME does not directly support renaming of baseline templates, so a means to provide version control and numbering was devised. This was done by exporting the baseline template, then editing the XML file, changing the name of the file and the name inside the XML, then reimporting the baseline template.

The configuration standard was broken into a series of baseline templates. These represent the common configurations that need to be verified, and that may change over time, and the more static configurations. To further prevent false positives from Cisco IOS Software rules being run on Catalyst® OS devices, some of the rules were designed to have a prerequisite called IOS12 defined, which can be used to prevent rules from matching and configurations being incorrectly applied to devices.

Table 1 lists the few samples of templates created by the customer.

Table 1. Customer Sample Templates

Template Name	Template Mode	Description of Template
SC_IOS_1_Syslog_Servers	Advanced	This template verifies that the correct syslog servers have been configured and checks for syslog servers that are not required. For example: + logging 172.20.24.5 + logging 172.20.24.68 + logging 172.30.24.68 - logging 172.30.24.29 - logging 172.20.24.44 - logging 172.20.24.2
SC_IOS_2_SNMP_Access	Advanced	This template checks for the list of access lists to be configured on the device. For example: +access-list 98 remark Permits subnets to RW SNMP +access-list 98 permit 172.22.103.0 0.0.0.255 +access-list 98 permit 172.2.24.0 0.0.0.255 +access-list 98 permit 172.3.24.0 0.0.0.255 +access-list 99 remark Permits subnets to RO SNMP +access-list 99 permit 172.20.103.0 0.0.0.255 +access-list 99 permit 172.20.24.0 0.0.0.255 +access-list 99 permit 172.30.24.0 0.0.0.255
SC_IOS_3_NTP	Advanced	This template checks whether the devices are configured with correct Network Time Protocol (NTP) server address for synchronisation of time. For example: +clock timezone [#.*#] +ntp server 172.20.152.20 prefer +ntp server 172.20.153.20 +ntp source [#.*#]
SC_IOS_4_Base_Global	Advanced	Checks for elements configured from the customer's base templates. For example: +snmp-server chassis-id [#.*#] -service udp-small-servers -service tcp-small-servers +service timestamps debug datetime msec +service timestamps log datetime msec +service sequence-numbers -logging trap notification -snmp-server enable traps syslog

SC_IOS_5_Base_Router	Advanced	The router Cisco IOS template verifies configuration for router-specific commands. For example: +router eigrp 1 +no service dhcp +no service pad +no ip bootp server Submode: interface [#.*Ethernet.*#] +no ip unreachablees +no ip redirects
SC_IOS_6_Base_Switch	Advanced	The switch Cisco IOS template verifies configuration for switch-specific commands. The configurations are not yet finalized by the company under study.
SC_IOS_7_Syslog_GIS_Server	Advanced	This template verifies that the correct syslog servers have been configured and checks for syslog servers that are not required. For example: +router eigrp 1 +no service dhcp +no service pad +no ip bootp server Submode: interface [#.*Ethernet.*#] +no ip unreachablees +no ip redirects

Case Study 2

Another retail customer of CiscoWorks was looking for a way to use policy templates to automate the process of conducting regular configuration checks on Cisco devices. In automating this, the customer would improve accuracy and consistency. Also this approach would be faster and more cost effective than the customer's former practice of manually performing configuration checks once a year.

Customer entities are connected to a worldwide wide area network (WAN) on Cisco routers that is provided by a global WAN (GWAN). The customer has implemented technologies like VPN, Multiprotocol Label Switching (MPLS), and so on across its network. One VPN is dedicated for data traffic (user traffic and server-to-server traffic), while the other is dedicated for management traffic only.

Data centers are connected to the GWAN through two points of presence (POP), providing MPLS connectivity. In each data center, several Cisco Catalyst 6513 Switches were installed in the core, distribution, and access layers of each block, totaling about 100 Cisco Catalyst 6513 Switches.

Need for Configuration and Change Management

As data centers are duplicated in the different theaters to serve local users and to provide load balancing and redundancy, data centers have become critical; therefore, configuration and change management have become a top priority for this customer.

In order to make sure that configurations are similar across the data centers, and to respect the defined policies around design, security, and best practices, the customer was scanning all the data center devices' configurations once a year, looking for exceptions to the organization's policies. This was a huge, repetitive task that was difficult to administer and prone to human errors.

Due to the large network size and distribution, it was not always possible to do the configuration checks successfully. If new devices were introduced in the network or if current network devices were changed, some of the devices might miss the default configuration that they should have.

Based on the fact that manual checks were difficult and effort consuming, it was done neither successfully nor completely. Hence the customer had to use policy templates to automatically enforce configurations across all data centers.

Solution

The use of the baseline compliance configuration available in CiscoWorks RME gives the following advantages to this customer:

- Periodically checks the existence or absence of required commands
- Creates detail reports on compliance deviation in Portable Document Format (PDF) and comma-separated value (CSV) format
- Lowers cost of maintenance, compared to a manual check
- Improves accuracy of results and increases frequency of checks
- Can adopt new features and support new devices
- Easy to report and deploy the missing commands
- Works on different devices (routers, switches, wireless)

CiscoWorks RME 4.x has features that perform these functions on the archive configurations managed by RME. In the baseline template, the export of compliance reports can be done manually. However, the customer required that the baseline template must have the ability to export those reports automatically (without manual intervention), on a periodic basis, to external locations where a third-party tool will collect the compliancy results. This feature is available in CiscoWorks LAN Management Solution RME 4.1 which part of LMS 3.0 release. (Examples of templates used by the company under study)

```
+ service timestamps debug datetime
+ service timestamps log datetime
+ service password-encryption
+ aaa authorization commands 1 default group tacacs+ local
+ aaa authorization commands 1 ConsoleLocal if-authenticated
+ aaa authorization commands 15 default group tacacs+ local
+ aaa authorization commands 15 ConsoleLocal if-authenticated
+ banner motd
*****
*****WARNING!!!This system is solely for the use of authorized users
for official purposes. You have no expectation of privacy in its use
and to ensure that the system is functioning properly, individuals
using this computer system are subject to having all of their
activities monitored and recorded by system personnel. Use of this
system evidences an express consent to such monitoring and agreement
that if such monitoring reveals evidence of possible abuse or criminal
activity, system personnel may provide the results of such monitoring
to appropriate
officials.*****
***** "
+ boot system flash sup-bootflash:s72033-ipervicesk9-mz.122-
18.SXF3.bin
+ boot system flash disk0:s72033-ipervicesk9-mz.122-18.SXF3.bin
SUBMODE
ip vrf SCZ4
```

```

+ description SCZ4 routing
+ rd [#.*:200#]
interface Loopback0
+ description --- For OSPF Router ID ---
+ ip address [#.*\.*\.*\.*\.*#] 255.255.255.255
interface Vlan101
+ description --- VLAN G1 ---
+ ip address [#.*\.*\.*\.*\.*#] 255.255.255.224
+ no ip redirects
+ no ip unreachable
+ no ip proxy-arp
- ip directed-broadcast
- ip mask-reply
+ ip ospf hello-interval 2
- ip ospf dead [#.*#]
+ ip ospf priority 0
+ ip ospf message-digest-key 1 md5 7 [#.*#]
- shutdown

```

Use Cases

The following use cases explain the concepts of working with a baseline template effectively.

Use Case 1: When there is more than one entry in the configuration files:

Commands in Device Config:

```

logging name1
logging name2
logging name3

```

Template: + logging [#!name1#] Result: Template is compliant

Explanation: This happens since the preceding template looks for a logging command in the device configuration but shouldn't have name1 as the logging server (since it is negated using !). The negation of name1 is done, which returns true, since there are other logging commands present with other names. So the template is compliant.

Use Case 2: When there is only one entry in the configuration file:

Commands in Device Config:

```

logging name1

```

Template: + logging [#!name1#] Result: Template is noncompliant

Explanation: The negation of name1 is done, which returns false. Since there is no other command with the logging statement except logging name1, the template is considered noncompliant.

Use Case 3: When there are no logging commands:

Commands in Device Config:

```
No logging commands
```

Template: + logging [#!name1#] Result: Template is noncompliant

Explanation: The negation of name1 is done first, which returns false. Since there are no logging commands, the template is noncompliant.

Additional Use Cases

Case 1: Compliance check on community strings

Problem Definition: The user wants to make sure that only two community strings are present in the configuration.

Configuration of any extra SNMP community strings in addition to the ones mentioned below leads to noncompliance.

Mandatory community strings in the device configuration:

```
snmp-server community white ro
snmp-server community black rw
```

Solution: Use of the negation concept to solve the problem. The template to solve this issue is:

```
- snmp-server community [#!white#] ro
- snmp-server community [#!black#] rw
```

In this case, the beginning dash character (-) will make sure that the command is not present in the device configuration, and since the negation is applied, it will match all the community strings except those mentioned in the problem definition and will flag a noncompliance if there are any community strings except the desired ones.

Running a compliance check would list the additional community strings present in the configuration. An alternate way could be to deploy the changes (removing the unwanted ones from the configuration) in the same job. Refer to Figure 1 for a better understanding.

Figure 1. Adding Job Details

The screenshot shows the 'Add Job Details' form in the Cisco Resource Manager Essentials interface. The form is titled 'Add Job Details' and is part of a 'Compliance Check' process. It includes sections for 'Scheduling', 'Job Info', and 'Job Options'. The 'Job Options' section has a checkbox for 'Check compliance and deploy' which is checked and circled in red. Other options include 'Copy Running Config to Startup' and 'Job Password'.


```
+ [#tacacs-server host 10.1.104.11 key.*#]
```

Since the tacacs-server host command is wrapped inside the regular expression, this would match any line that has the TACACS server host 10.1.104.11 key. This is the correct template to match the tacacs-server host command in the device configuration.

Case 4: Replacing helper addresses

Problem Definition: The customer has helper addresses configured on routers, under various interfaces (E0 on router1, E1 on router2, E0/1 on router3...) and wants to replace those with new addresses.

For example:

```
IF there is (in the configuration)
IP helper-address 1.1.1.1
THEN replace it with
IP helper-address 2.2.2.2
```

Solution: This can be achieved using compliance management with baseline templates. As discussed earlier in this document, prerequisite templates can be used to solve this problem. You can have a prerequisite command set that will check for the IF condition, and if the condition matches, then the THEN part would execute.

Case 4(a) Replacing IP helper addresses on all interfaces

This is how the prerequisite template looks. This will check whether IP helper-address 1.1.1.1 is configured in all the Ethernet interfaces.

Name: CheckIpHelperAddress

Submode: Yes

Is Prerequisite: Yes

Ordered: No

Prerequisite-Commandset: none

Parent: none

```
interface [#Ethernet.*#]
+ ip helper-address 1.1.1.1
```

In the preceding template, interface [#Ethernet.*#] is the submode command, as the IP helper address 1.1.1.1 applies to interfaces, and you will have to go to the interface configuration mode to check for the same. This template is marked as Prerequisite.

Now create another template to replace IP helper-address 1.1.1.1 with IP helper-address 2.2.2.2.

Name: ReplacelpAddress

Submode: Yes

Is Prerequisite: No

Ordered: No

Prerequisite-Commandset: CheckIpHelperAddress

Parent: none

```
interface [#Ethernet.*#]
- ip helper-address 1.1.1.1
+ ip helper-address 2.2.2.2
```

Refer to Figures 2 and 3 for a better understanding of the solution

Figure 2. Adding Template Details

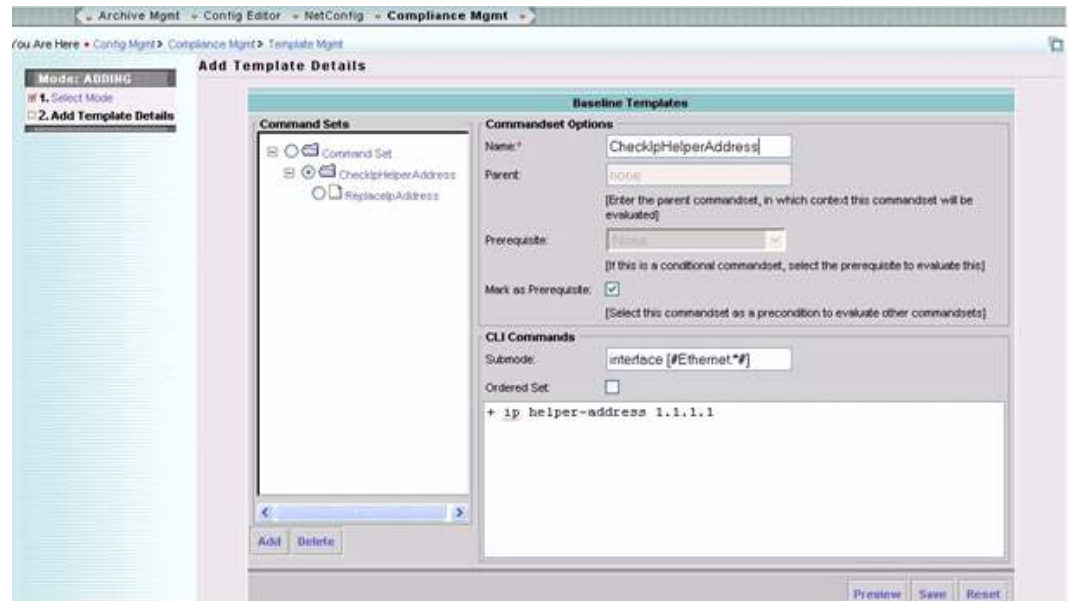
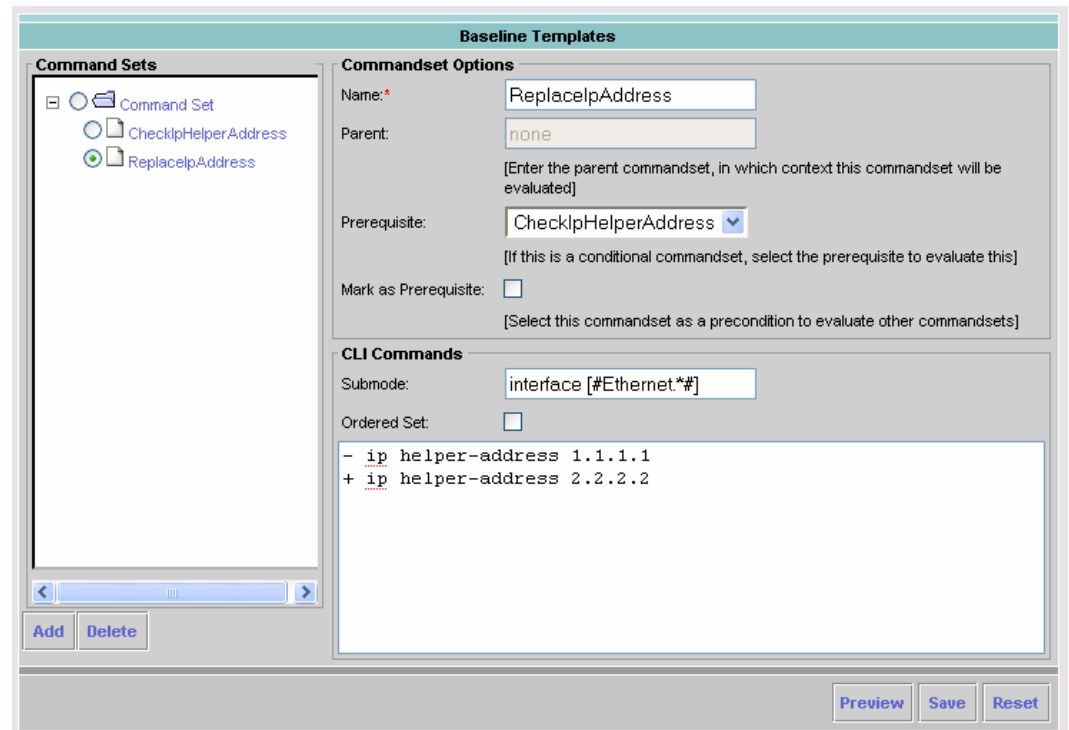


Figure 3. Solving the Helper Address Problem



The ReplacelpAddress template has CheckIpHelperAddress as its prerequisite. Running a Compliance and Deploy would scan through all the Ethernet interfaces against the selected devices, and if IP helper-address 1.1.1.1 is configured in any of the Ethernet interfaces, all the Ethernet interfaces will be configured with IP helper-address 2.2.2.2. The prerequisite template is considered compliant if IP helper-address 1.1.1.1 is present in any one of the Ethernet interfaces.

Case 4(b): Replacing IP helper addresses only on matching addresses using an advanced template

Deploying the changes only to those interfaces that have the IP helper-address 1.1.1.1 configured is also possible with the CiscoWorks RME baseline template. In this scenario, the template structure should be as follows:

Name: CheckIpHelperAddress

SubMode: Yes

IsPrerequisite: Yes

Ordered: No

Prerequisite-Commandset: none

Parent: none

```
interface [#Ethernet.*#]
+ ip helper-address 1.1.1.1
```

Name: ReplacelpAddress

SubMode: No

IsPrerequisite: No

Ordered: No

Prerequisite-Commandset: CheckIpHelperAddress

Parent: CheckIpHelperAddress

```
- ip helper-address 1.1.1.1
+ ip helper-address 2.2.2.2
```

Refer to Figures 4 and 5 for a better understanding.

Figure 4. Deploying Changes to Specific Interfaces

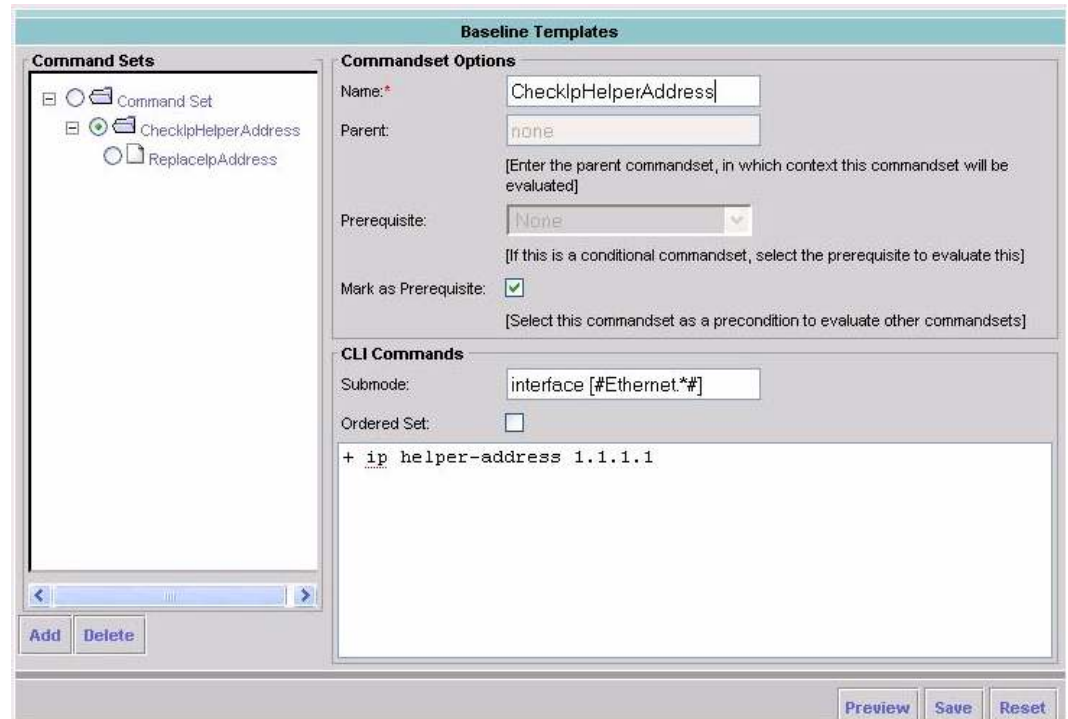
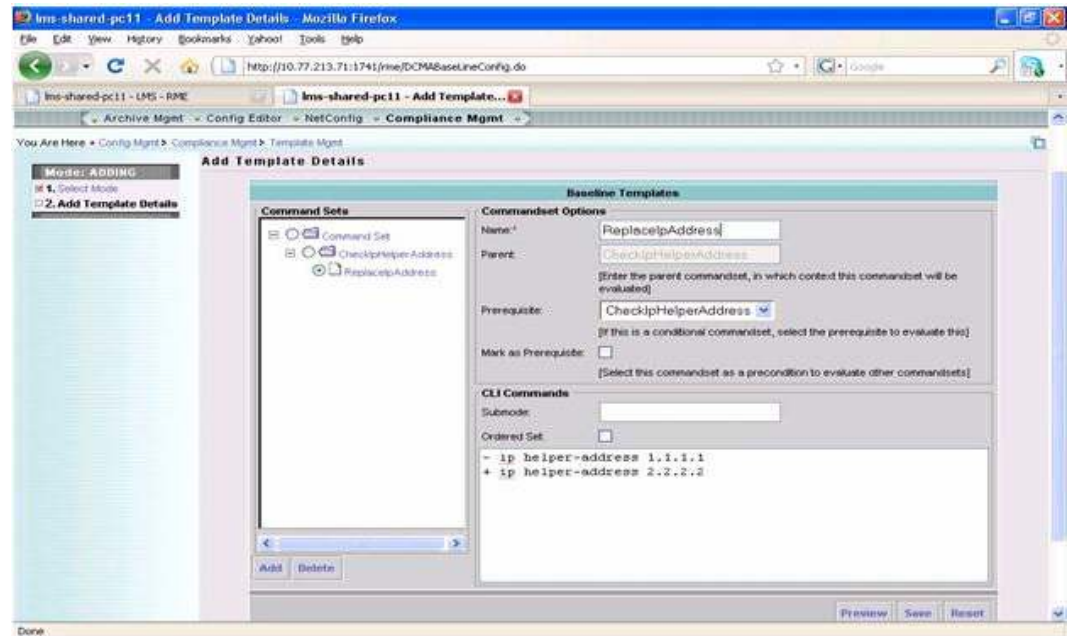


Figure 5. Solving Case 4(b)



In this case, the parent command set should be a prerequisite for the child command set, so the child will be executed under the submode of the parent. Hence a Deploy can be done to the Ethernet interfaces that have IP helper-address 1.1.1.1 configured in it.

Case 5: Adding a default configuration set on shutdown interfaces using the basic template

With LMS 3.1 and RME 4.2, enhancements are done to the compliance management module, which enables users to do the use case mentioned in 4(b) even with the basic templates instead of using the advanced template. For instance, let us take another example similar to case 4(b) as stated below.

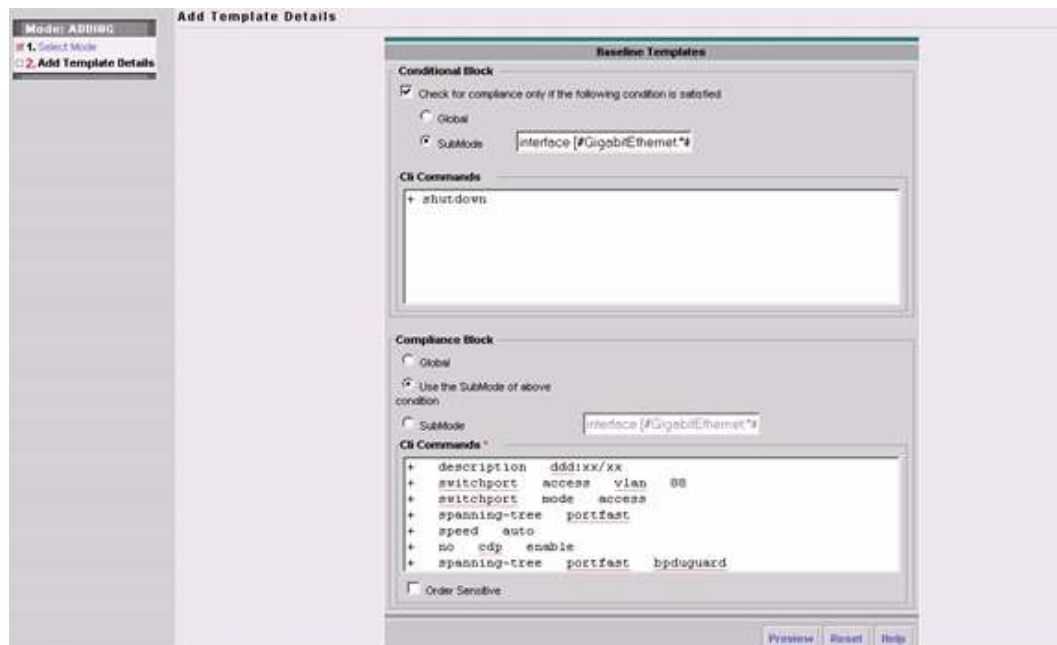
Problem Definition: Look into the state of all Gigabit Ethernet interfaces in a given device and apply a set of default configurations on those interfaces that are administratively shut down.

Default configuration to be applied:

```
description ddd:xx/xx
switchport access vlan 88
switchport mode access
spanning-tree portfast
speed auto
no cdp enable
spanning-tree portfast bpduguard
```

Solution: In the basic template, the user has to provide the state of the interface in the conditional block. Specify that +shutdown should be checked in the interface submode (see Figure 6). This can be done by selecting the option submode and providing the regular expression as interface [#GigabitEthernet.*#]. While running the job, this checks for the preconditional block to be true. When this condition is satisfied, the commands specified in the compliance block will be checked. Choose the option “Use the submode of above” to check the default configuration in the corresponding interface for which the conditional block is true.

Figure 6. Checking the State of the Interface in the Conditional Block



Case 6: Compliance check for routing protocol configurations

Problem Definition: The customer is trying to set up a compliance check for network statements under Enhanced Interior Gateway Routing Protocol (EIGRP). The user is trying to get it to match

numerous IP addresses that are used at the remote site. All the addresses need to match the following format:

```
network 10.XX.X [1-9].0 0.0.0.255
```

Solution: Regular expressions can be used to match the requirement. In the preceding scenario, the requirement is fairly straightforward, except to match the 10.XX.X [1-9].0. The expression `\d` can be used to match any digit in regular expressions. With this, the regular expression would be like:

```
10\.\d\d\.\d[1-9]\.0
```

Where `\d` would match any digit from 0 through 9.

Anything between the square brackets in a regular expression would match only one character or a digit. Specifying `[1-9]` matches anything between 1 and 9, both inclusive, so the template to match the user's requirement is:

```
+ network [#10\.\d\d\.\d[1-9]\.0#] 0.0.0.255
```

Running a Compliance Check on More Than One Interface

This section describes how to create a baseline template to check 2950 switches for the interfaces without port security while having port security enabled for all ports.

```
Submode Command: interface [#.*#]
```

CLI commands:

```
+ switchport port-security
```

Running a compliance check on all the interfaces would give a report on the interfaces that do not have the switch port security configured. This can then be deployed to the noncompliant interfaces. In a nutshell, to make any change or to check for compliance against multiple/all interfaces, a submode command should be provided as follows:

```
Interface [#.*#] Run against all the interfaces in the device.
```

```
Interface [#Fast.*#] Run against all the interfaces starting with Fast, typically on all Fast Ethernet Interfaces.
```

```
Interface [#Ethernet.*#] On all Ethernet interfaces, and so on.
```

This can be extended to any level. Assume you want to turn on quality of service (QoS) autodiscovery on all the VLAN interfaces. Provide a submode command to enter into the VLAN interface and then the configuration command to enable autodiscovery. Since you want to enable the discovery on all the VLAN interfaces, the baseline template should run on all the VLAN interfaces. So, use the wildcard to specify that it should be run on all VLAN interfaces. The submode command (command to get into the VLAN interfaces) should be:

```
+ interface [#Vlan.*#] Applies to all VLAN interfaces
```

The CLI command to enable the autodiscovery of QoS should be:

```
+ auto discovery qos
```

You could either run a compliance to see if the autodiscovery is already enabled or do compliance and deploy as a single job by selecting the Compliance and Deploy option in the Compliance Job flow.

If you need to disable autodiscovery on all these interfaces at some later point of time, the CLI command would be:

```
- auto discovery qos
```

This will help ensure that autodiscovery is not enabled on all VLAN interfaces, and even if it is enabled, the baseline template will disable autodiscovery by running no autodiscovery QoS.

Compliance of Encrypted/Hashed Passwords

It is common to have devices configured with the option Enable Secret Passwords. Running a compliance job to match the Enable Secret Passwords option is a little tricky. Irrespective of whether you provide an encrypted or unencrypted password, the Enable Secret Passwords that appear in the device configuration are always encrypted. To be more specific, the passwords are hashed using Message Digest Algorithm 5 (MD5). A typical example of an enable secret appearing in the device configurations is:

```
enable secret 5 $1$zAdO$2PDn1Ia8XmF1lS5muu3i1.
```

In order to match particular enable secret password/run compliance, you will have to provide the encrypted password in the baseline template and not the unencrypted one. The enable secret in the preceding example is secret, and if you have a template with an unencrypted password like:

```
+ enable secret lab
```

it will not match the enable secret password in the devices. You have to provide the encrypted password in the template to match correctly, so the correct template would be:

```
+ enable secret 5 $1$zAdO$2PDn1Ia8XmF1lS5muu3i1.
```

In case of noncompliance, you could simply deploy the command as it is to make it compliant.

Multiline Command Compliance in Baseline Templates

Baseline templates can match a multiline command (like multiline banner commands) as well.

Suppose that you have a multiline banner command like:

```
banner motd "
=====
*****WARNING*****
=====
"
```

<NL> is the tag used to match the new line character, so the template to match this sort of a multiline banner command would be:

```
+ banner motd
"<NL>=====<NL>*****WARNING*****<NL>=====<NL>
L>"
```

Baseline templates are handy for multiline comparison as well.

Case 7: Search and replace

Problem Definition: The requirement is to make a global modification on all device configurations from an old IP address to a new IP address. For example, you want to change the following configuration statements from the IP address 1.1.1.1 to 1.1.1.2:

```
permit tcp host 1.1.1.1 any equ telnet
logging 1.1.1.1
snmp-server host 1.1.1.1 pcode
```

```
snmp-trap host 1.1.1.1 pcode
```

Solution: There have been similar problems and solutions in earlier case studies. A search and replace feature, as in any text editor, can be used to resolve this problem. A baseline template to search for these configurations in the global configuration and then replace them with the new one would be as simple as:

```
- permit tcp host 1.1.1.1 any equ telnet
- logging 1.1.1.1
- snmp-server host 1.1.1.1 pcode
- snmp-trap host 1.1.1.1 pcode
+ permit tcp host 1.1.1.2 any equ telnet
+ logging 1.1.1.2
+ snmp-server host 1.1.1.2 pcode
+ snmp-trap host 1.1.1.2 pcode
```

Remove all the configuration commands that have the old IP address with the – option and then add the new commands that need to be present in the configuration (with the new IP address).

Specify the importance of the order of the commands in the device configuration.

Case 8: Using an ordered set to check for ACL compliance

Problem Definition: The user has a class A network 36.0.0.0 and its subnet mask is 255.255.0.0. The user wants to allow access to one address on subnet 48 and reject all others on that subnet. Additionally, the user would also like to accept traffic on all other network 36.0.0.0 subnets.

Solution: Packet filtering helps control packet movement through the network. Such control can help limit network traffic and restrict network use by certain users or devices. To permit or deny packets from crossing specified interfaces, access lists are provided. The ordering of the ACLs is very critical, as they are evaluated in the order of their appearance in the device configuration. Once a match is found, the rest are ignored. So to check for a proper ACL, you will have to consider the order of commands as well. The ordered set is a useful feature in baseline templates through which this problem could be solved. A typical ACL for this given problem would be:

```
access-list 2 permit 36.48.0.3
access-list 2 deny 36.48.0.0 0.0.255.255
access-list 2 permit 36.0.0.0 0.255.255.255
```

Template to check for this compliance is:

```
+ access-list 2 permit 36.48.0.3
+ access-list 2 deny 36.48.0.0 0.0.255.255
+ access-list 2 permit 36.0.0.0 0.255.255.255
```

An ordered set should be selected for a proper compliance.

```
+ access-list 2 permit 36.48.0.3
```

The preceding template command provides a strict matching. However, you could use a range of addresses through regular expressions. Suppose you wanted to allow any address between 2 to 15 in the 48 subnet; then you could modify the preceding command as:

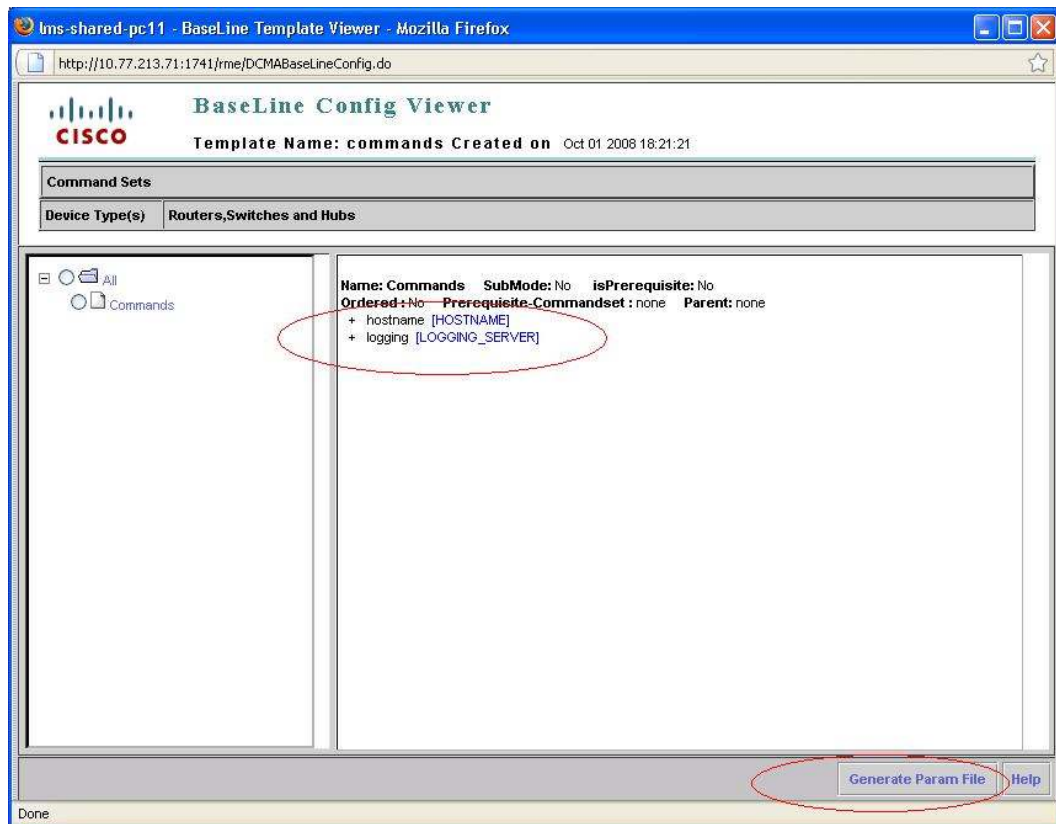
```
+ access-list 2 permit 36.48.0.[#[2-15]#]
```

Case 9: Deploy using an XML parameter file

Problem Definition: The user wants to use this feature heavily in the commissioning of new devices that have been preloaded with a minimal configuration. The need is to use parameter files and build a logic for mathematical variations such as IP addresses/hostnames, and so on.

Solution: This can be done using the Variable parameters option in the baseline. For example, you can create a basic template with the Variable parameters. Refer to Figure 7 for a better understanding.

Figure 7. Working with Devices



Use the Generate Param File button to generate the XML file. Before you deploy the template, the XML file needs to be updated with the device names and the parameter values. For example, refer to the following XML details:

```
<?xml version="1.0" encoding="UTF-8"?>
<ConfigData>
<Device HostName="10.77.210.168">
<DataSet CommandletName="Commands">
<Param>
<Name>HOSTNAME</Name>
<Value>10.77.210.168</Value>
</Param>
<Param>
<Name>LOGGING_SERVER</Name>
<Value>10.1.1.1</Value>
</Param>
</DataSet>
</Device>
<Device HostName="10.77.210.61">
<DataSet CommandletName="Commands">
<Param>
<Name>HOSTNAME</Name>
<Value>10.77.210.61</Value>
</Param>
<Param>
<Name>LOGGING_SERVER</Name>
<Value>10.1.1.2</Value>
</Param>
</DataSet>
</Device>
</ConfigData>
```

Hence different values can be given to the hostname and logging server for each device. Refer to the section “How to deploy using the parameterized file” in the document “CiscoWorks Resource Manager Essentials Compliance Management” for more details. See Table 2.

Table 2. Other General Examples

If You Want To...	Then You Should...
<p>Use the baseline template to match either of these two trap destinations, 10.0.8.201 and 10.0.8.202. The expression should match both the following commands:</p> <pre>"snmp-server host 10.0.8.201 public" "snmp-server host 10.0.8.202 public"</pre>	<p>Use the command "+ snmp-server host [#10.0.8.20[12]#] public" in the template to take care of the requirement.</p>
<p>Use a baseline expression that matches the command "+ set trunk 2/1 nonegotiate dot1q 12,300,1025-4094" (the ID "12" present in the command can vary) even if the 12 is changed to any other number.</p>	<p>Use the command "+ set trunk 2/1 nonegotiate dot1q [#(\d+),300,1025-4094#]" in the template</p>

FAQ

Q. How/where does the configuration get deployed on the router? Is there any transport protocol that allows the baseline template to be applied to the startup configuration?

A. The baseline template is always deployed to the running configuration. Currently, there is no option that would copy the template into the startup configuration directly. However, the CiscoWorks RME job can save the changes made to the running configuration into the startup configuration by enabling Copy Running Config to Startup in the Job Options. Refer to the following link on deploying a baseline template:

http://cisco.com/en/US/prod/collateral/netmgtsw/ps6504/ps6528/ps2073/prod_white_paper0900aecd8068cc98.pdf.

Q. Can the baseline template be used to overwrite the device configuration so that there are no other commands in the device configuration other than the ones in the baseline template?

A. No. This cannot be done using a baseline template.

Q. Why is the solution to a Deploy job failing for most of the commands with the message Invalid input detected at the “^” marker?

A. Make sure that the command is deployed/checked for compliance in the proper mode. For instance, interface-related configurations cannot be deployed or expected to be present in the global configuration mode. This error indicates that the command is deployed in the wrong mode. The problem is due to the absence of the submode command. Provide a proper submode command and try again.

Q. What protocol does a baseline template use to deploy the configuration change?

A. It uses the protocol order defined for the Archive Management “Config Deploy”. This protocol order can be changed by selecting Admin >> Config Mgmt >> Transport Settings >> Archive Mgmt (Application Name).

Conclusion

With compliance management tools like baseline templates in place, customers could attain the following business values:

- Cost savings
 - Reduced operating expenditures
 - Improved efficiency – Utilization of configuration leading practices
- Productivity
 - 5000 device CiscoWorks implementation
 - Automated change and configuration management process
 - Performance optimization
- Risk management and compliance
 - Better adherence to industry and regulator compliance
 - Improved visibility of network infrastructure

The feature will help organizations automate a network configuration audit process that produces a detailed report of the devices and changes required to make the network compliant to defined policy. These required changes can be fed into the change approval process and upon approval automatically pushed out to the network and executed as network configuration changes. The end result could allow users to automate several of the more time-consuming tasks in enforcing and maintaining configuration policy on the network, allowing them to improve network stability and performance without increasing operating expenditures.



Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

CCDE, CCSI, CCENT, Cisco Eos, Cisco HealthPresence, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco Nurse Connect, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCI, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0903R)