Cisco Networking Services Configuration Guide, Cisco IOS Release 12.2SR

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Last Updated: August 09, 2011

and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Configuring Cisco Networking Services

The Cisco Networking Services (CNS) feature is a collection of services that can provide remote event-driven configuring of Cisco IOS networking devices and remote execution of some command-line interface (CLI) commands.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Cisco Networking Services

- Configure the remote router to support the Cisco Networking Services configuration agent and the Cisco Networking Services event agent.
- Configure a transport protocol on the remote router that is compatible with the remote router's external interface. The following table lists the supported transport protocols that can be used depending on the router interface.
- Create the configuration template in the Cisco Networking Services configuration-engine provisioning database. (This task is best done by a senior network designer.)

Table 1 Router Interface and Transport Protocols Required by Cisco Networking Services Services

	Transport Protocol		
Router Interface	SLARP	ATM InARP	PPP (IPCP)
T1	Yes	Yes	Yes
ADSL	No	Yes	Yes
Serial	Yes	No	Yes

Cisco Networking Services Image Agent

- Determine where to store the Cisco IOS images on a file server to make the image available to many other networking devices. If the Cisco Networking Services Event Bus is to be used to store and distribute the images, the Cisco Networking Services event agent must be configured.
- Set up a file server to enable the networking devices to download the new images. Protocols such as TFTP, HTTP, HTTPS, and rcp can be used.
- Determine how to handle error messages generated by Cisco Networking Services image agent operations. Error messages can be sent to the Cisco Networking Services Event Bus or an HTTP or HTTPS URL.

Restrictions for Cisco Networking Services

Cisco Networking Services Configuration Engine

- The Cisco Networking Services configuration engine must be the Cisco Intelligence Engine 2100 (Cisco IE2100) series and must be running software version 1.3.
- The configuration engine must have access to an information database of attributes for building a configuration. This database can reside on the Cisco IE2100 itself.
- Configuration templates must be prepared on the Cisco Networking Services configuration engine before installation of the remote router.
- The user of Cisco Networking Services Flow-Through Provisioning and the Cisco Networking Services configuration engine must be familiar with designing network topologies, designing configuration templates, and using the Cisco Networking Services configuration engine.

Cisco Networking Services Image Agent

During automated image loading operations you must try to prevent the Cisco IOS device from losing connectivity with the file server that is providing the image. Image reloading is subject to memory issues and connection issues. Boot options must also be configured to allow the Cisco IOS device to boot another image if the first image reload fails. For more details see the “Managing Configuration Files” module of the *Cisco IOS Configuration Fundamentals Configuration Guide* .

Cisco Networking Services Frame Relay Zero Touch

The Cisco Networking Services Frame Relay Zero Touch solution does not support switched virtual circuits (SVCs).

The Frame Relay zero touch solution does not support IP over PPP over Frame Relay because routing to an interface (or subinterface) that supports IP over PPP over Frame Relay is not possible.

Command Scheduler

The EXEC CLI specified in a Command Scheduler policy list must neither generate a prompt nor can it be terminated using keystrokes. Command Scheduler is designed as a fully automated facility, and no manual intervention is permitted.

Remote Router

- The remote router must run a Cisco IOS image that supports the Cisco Networking Services configuration agent and Cisco Networking Services event agent.
- Ports must be prepared on the remote router for connection to the network.
- You must ensure that the remote router is configured using Cisco Configuration Express.

Information About Cisco Networking Services

Cisco Networking Services

Cisco Networking Services is a foundation technology for linking users to networking services and provides the infrastructure for the automated configuration of large numbers of network devices. Many IP networks are complex with many devices, and each device must currently be configured individually. When standard configurations do not exist or have been modified, the time involved in initial installation and subsequent upgrading is considerable. The volume of smaller, more standardized, customer networks is also growing faster than the number of available network engineers. Internet service providers (ISPs) now

need a method for sending out partial configurations to introduce new services. To address all these issues, Cisco Networking Services has been designed to provide “plug-and-play” network services using a central directory service and distributed agents. Cisco Networking Services features include Cisco Networking Services configuration and event agents and a Flow-Through Provisioning structure. The configuration and event agents use a Cisco Networking Services configuration engine to provide methods for automating initial Cisco IOS device configurations, incremental configurations, and synchronized configuration updates, and the configuration engine reports the status of the configuration load as an event to which a network monitoring or workflow application can subscribe. The Cisco Networking Services Flow-Through Provisioning uses the Cisco Networking Services configuration and event agents to provide an automated workflow, eliminating the need for an on-site technician.

Cisco Networking Services Configuration Agent

The Cisco Networking Services configuration agent is involved in the initial configuration and subsequent partial configurations on a Cisco IOS device. To activate the Cisco Networking Services configuration agent, enter any of the **cns config** CLI commands.

Initial Cisco Networking Services Configuration

When a routing device first comes up, it connects to the configuration server component of the Cisco Networking Services configuration agent by establishing a TCP connection through the use of the **cns config initial**command, a standard CLI command. The device issues a request and identifies itself by providing a unique configuration ID to the configuration server.

When the Cisco Networking Services web server receives a request for a configuration file, it invokes the Java servlet and executes the corresponding embedded code. The embedded code directs the Cisco Networking Services web server to access the directory server and file system to read the configuration reference for this device (configuration ID) and template. The Configuration Agent prepares an instantiated configuration file by substituting all the parameter values specified in the template with valid values for this device. The configuration server forwards the configuration file to the Cisco Networking Services web server for transmission to the routing device.

The Cisco Networking Services configuration agent accepts the configuration file from the Cisco Networking Services web server, performs XML parsing, checks syntax (optional), and loads the configuration file. The routing device reports the status of the configuration load as an event to which a network monitoring or workflow application can subscribe.

For more details on using the Cisco Cisco Networking Services configuration engine to automatically install the initial Cisco Networking Services configuration, see the *Cisco Networking Services Configuration Engine Administrator’s Guide* at http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cns/ce/rel13/ag13/index.htm .

Incremental Cisco Networking Services Configuration

Once the network is up and running, new services can be added using the Cisco Networking Services configuration agent. Incremental (partial) configurations can be sent to routing devices. The actual configuration can be sent as an event payload by way of the event gateway (push operation) or as a signal event that triggers the device to initiate a pull operation.

The routing device can check the syntax of the configuration before applying it. If the syntax is correct, the routing device applies the incremental configuration and publishes an event that signals success to the configuration server. If the device fails to apply the incremental configuration, it publishes an event that indicates an error.

Once the routing device has applied the incremental configuration, it can write the configuration to NVRAM or wait until signaled to do so.

Synchronized Configuration

When a routing device receives a configuration, the device has the option to defer application of the configuration upon receipt of a write-signal event. The Cisco Networking Services Configuration Agent feature allows the device configuration to be synchronized with other dependent network activities.

Cisco Networking Services Config Retrieve Enhancement with Retry and Interval

The Cisco Networking Services Config Retrieve Enhancement with Retry and Interval feature adds new functionality to the **cns config retrieve** command enabling you to specify the retry interval and an amount of time in seconds to wait before attempting to retrieve a configuration from a trusted server.

Cisco Networking Services EXEC Agent

The CNS EXEC agent allows a remote application to execute an EXEC mode CLI command on a Cisco IOS device by sending an event message that contains the command. A restricted set of EXEC **show** commands is supported.

Cisco Networking Services Event Agent

Although other Cisco Networking Services agents may be configured, no other Cisco Networking Services agents are operational until the **cns event**command is entered because the Cisco Networking Services event agent provides a transport connection to the Cisco Networking Services event bus for all other Cisco Networking Services agents. The other Cisco Networking Services agents use the connection to the Cisco Networking Services event bus to send and receive messages. The Cisco Networking Services event agent does not read or modify the messages.

Cisco Networking Services Image Agent

Administrators maintaining large networks of Cisco IOS devices need an automated mechanism to load image files onto large numbers of remote devices. Existing network management applications are useful to determine which images to run and how to manage images received from the Cisco online software center. Other image distribution solutions do not scale to cover thousands of devices and cannot distribute images to devices behind a firewall or using Network Address Translation (NAT). The Cisco Networking Services image agent enables the managed device to initiate a network connection and request an image download allowing devices using NAT, or behind firewalls, to access the image server.

The Cisco Networking Services image agent can be configured to use the Cisco Networking Services Event Bus. To use the Cisco Networking Services Event Bus, the Cisco Networking Services event agent must be enabled and connected to the Cisco Networking Services event gateway in the Cisco Networking Services Configuration Engine. The Cisco Networking Services image agent can also use an HTTP server that understands the Cisco Networking Services image agent protocol. Deployment of Cisco Networking Services image agent operations can use both the Cisco Networking Services Event Bus and an HTTP server.

Cisco Networking Services Results Messages

When a partial configuration has been received by the router, each line of the configuration will be applied in the same order as it was received. If the Cisco IOS parser has an error with one of the lines of the configuration, then all the configuration up to this point will be applied to the router, but none of the configuration beyond the error will be applied. If an error occurs, the **cns config partial**command will retry until the configuration successfully completes. In the pull mode, the command will not retry after an error. By default, NVRAM will be updated except when the **no-persist** keyword is configured.

A message will be published on the Cisco Networking Services event bus after the partial configuration is complete. The Cisco Networking Services event bus will display one of the following status messages:

- cisco.mgmt.cns.config.complete--Cisco Networking Services configuration agent successfully applied the partial configuration.
- cisco.mgmt.cns.config.warning--Cisco Networking Services configuration agent fully applied the partial configuration, but encountered possible semantic errors.
- cisco.mgmt.cns.config.failure(CLI syntax)--Cisco Networking Services configuration agent encountered a command line interface (CLI) syntax error and was not able to apply the partial configuration.
- cisco.mgmt.cns.config.failure(CLI semantic)--Cisco Networking Services configuration agent encountered a CLI semantic error and was not able to apply the partial configuration.

In Cisco IOS Releases 12.4(4)T, 12.2 (33)SRA, and later releases, a second message is sent to the subject “cisco.cns.config.results” in addition to the appropriate message above. The second message contains both overall and line-by-line information about the configuration that was sent and the result of the action requested in the original message. If the action requested was to apply the configuration, then the information in the results message is semantic in nature. If the action requested was to check syntax only, then the information in the results message is syntactical in nature.

Cisco Networking Services Message Formats

SOAP Message Format

Using the Service-Oriented Access Protocol (SOAP) protocol provides a way to format the layout of Cisco Networking Services messages in a consistent manner. SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses extensible markup language (XML) technologies to define an extensible messaging framework that provides a message format that can be exchanged over a variety of underlying protocols.

Within the SOAP message structure, there is a security header that enables Cisco Networking Services notification messages to authenticate user credentials.

Cisco Networking Services messages are classified into three message types: request, response and notification. The formats of these three message types are defined below.

Request Message

The following is the format of a Cisco Networking Services request message to the Cisco IOS device:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="0">
```

```
      <wsse:usernameToken>
        <wsse:Username>john</wsse:Username>
        <wsse:Password>cisco</wsse:Password>
      </wsse:usernameToken>
    </wsse:Security>
    <cns:cnsHeader version="1.0" xmlns:cns="http://www.cisco.com/management/cns/envelope">
      <cns:Agent>CNS_CONFIG</cns:Agent>
      <cns:Request>
        <cns:correlationID>IDENTIFIER</cns:correlationID>
        <cns:ReplyTo>
          <cns:URL>http://10.1.36.9:80/cns/ResToServer</cns:URL>
        </cns:ReplyTo>
      </cns:Request>
      <cns:Time>2003-04-23T20:27:19.847Z</cns:Time>
    </cns:cnsHeader>
  </SOAP:Header>
  <SOAP:Body xmlns="http://www.cisco.com/management/cns/config">
    <config-event config-action="read" no-syntax-check="TRUE">
      <config-data>
        <config-id>AAA</config-id>
        <cli>access-list 1 permit any</cli>
      </config-data>
    </config-event>
  </SOAP:Body>
</SOAP:Envelope>
```

Note The ReplyTo field is optional. In the absence of the ReplyTo field, the response to the request will be sent to the destination where the request originated. The body portion of this message contains the payload and is processed by the Cisco Networking Services agent mentioned in the Agent field.

Response Message

The following is the format of a Cisco Networking Services response message from the Cisco IOS device as a response to a request:

```
?xml version="1.0" encoding="UTF-8"?
SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
SOAP:Header
wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="true"
wsse:UsernameToken
wsse:Username infysj-7204-8 /wsse:Username
wsse:Password NTM3NTg2NzIzOTg2MTk2MjgzNQ==/wsse:Password
/wsse:UsernameToken /wsse:Security
CNS:cnsHeader Version="2.0" xmlns:CNS="http://www.cisco.com/management/cns/envelope"
CNS:Agent CNS_CONFIG /CNS:Agent
CNS:Response
CNS:correlationID IDENTIFIER /CNS:correlationID
/CNS:Response
CNS:Time 2005-06-23T16:27:36.185Z /CNS:Time
/CNS:cnsHeader
/SOAP:Header
SOAP:Body xmlns="http://www.cisco.com/management/cns/config"
config-success config-id AAA /config-id /config-success
/SOAP:Body
/SOAP:Envelope
```

Note The value of CorrelationId is echoed from the corresponding request message.

The body portion of this message contains the response from the Cisco IOS device to a request. If the request is successfully processed, the body portion contains the value of the response put in by the agent that processed the request. If the request cannot be successfully processed, then the body portion will contain an error response.

Notification Message

The following is the format of a Cisco Networking Services notification message sent from the Cisco IOS device:

```
?xml version="1.0" encoding="UTF-8"?
SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
SOAP:Header
wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="true"
wsse:UsernameToken
wsse:Username dvlpr-7200-2 /wsse:Username
wsse:Password /wsse:Password
/wsse:UsernameToken
/wsse:Security
CNS:cnsHeader version="2.0" xmlns:CNS="http://www.cisco.com/management/cns/envelope"
CNS:Agent CNS_CONFIG_CHANGE/CNS:Agent
CNS:Notify /CNS:Notify
CNS:Time 2006-01-09T18:57:08.441Z/CNS:Time
/CNS:cnsHeader
/SOAP:Header
SOAP:Body xmlns="http://www.cisco.com/management/cns/config-change"
configChanged version="1.1" sessionData="complete"
sequence lastReset="2005-12-11T20:18:39.673Z" 7 /sequence
changeInfo
user/user
async port con_0 /port /async
when
absoluteTime 2006-01-09T18:57:07.973Z /absoluteTime
/when
/changeInfo
changeData
changeItem
context /context
enteredCommand
cli access-list 2 permit any /cli
/enteredCommand
oldConfigState
cli access-list 1 permit any /cli
/oldConfigState
newConfigState
cli access-list 1 permit any /cli
cli access-list 2 permit any /cli
/newConfigState
/changeItem
/changeData
/configChanged
/SOAP:Body
/SOAP:Envelope
```

A notification message is sent from the Cisco IOS device without a corresponding request message when a configuration change is made. The body of the message contains the payload of the notification and it may also contain error information. If the request message sent to the Cisco IOS device fails in XML parsing and the CorrelationId field cannot be parsed, then an error notification message will be sent instead of an error response.

Error Reporting

Error is reported in the body of the response or a notification message in the SOAP Fault element. The following is the format for reporting errors.

```
?xml version="1.0" encoding="UTF-8"?
SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
SOAP:Header
wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="true"
wsse:UsernameToken
wsse:Username dvlpr-7200-2 /wsse:Username
wsse:Password /wsse:Password
```

```
/wsse:UsernameToken
/wsse:Security
CNS:cnsHeader version="2.0" xmlns:CNS="http://www.cisco.com/management/cns/envelope"
CNS:Agent CNS_CONFIG /CNS:Agent
CNS:Response
CNS:correlationID SOAP_IDENTIFIER /CNS:correlationID
/CNS:Response
CNS:Time 2006-01-09T19:10:10.009Z /CNS:Time
/CNS:cnsHeader
/SOAP:Header
SOAP:Body xmlns="http://www.cisco.com/management/cns/config"
SOAP:Detail
config-failure
config-id AAA /config-id
error-info
line-number 1 /line-number
error-message CNS_INVALID_CLI_CMD /error-message
/error-info
/config-failure
/SOAP:Detail
/SOAP:Fault
/SOAP:Body
/SOAP:Envelope
```

Cisco Networking Services Security Enhancement

Before the introduction of the Cisco Networking Services Security Enhancement feature, the Cisco Networking Services message format did not support security. Using the new Cisco Networking Services SOAP message structure, the username and password are authenticated.

If authentication, authorization, and accounting (AAA) is configured, then Cisco Networking Services SOAP messages will be authenticated with AAA. If AAA is not configured, there will be no authentication. For backward compatibility, Cisco Networking Services will support the existing non-SOAP message format and will respond accordingly without security.

The **cns aaa authentication**command is required to turn on Cisco Networking Services Security Enhancement. This command determines whether the Cisco Networking Services messages are using AAA security or not. If the **cns aaa authentication**command is configured, then all incoming SOAP messages into the device are authenticated by AAA.

Cisco Networking Services Interactive CLI

The Cisco Networking Services Interactive CLI feature provides a XML interface that allows you to send interactive commands to a router, such as commands that generate prompts for user input. A benefit of this feature is that interactive commands can be aborted before they have been fully processed. For example, for commands that generate a significant amount of output, the XML interface can be customized to limit the size of the output or the length of time allowed for the output to accumulate. The capability to use a programmable interface to abort a command before its normal termination (similar to manually aborting a command) can greatly increase the efficiency of diagnostic applications that might use this functionality. The new XML interface also allows for multiple commands to be processed in a single session. The response for each command is packaged together and sent in a single response event.

Cisco Networking Services IDs

The Cisco Networking Services ID is a text string that is used exclusively with a particular Cisco Networking Services agent. The Cisco Networking Services ID is used by the Cisco Networking Services agent to identify itself to the server application with which it communicates. For example, the Cisco Networking Services configuration agent will include the configuration ID when communicating between the networking device and the configuration server. The configuration server uses the Cisco Networking

Services configuration ID as a key to locate the attribute containing the Cisco IOS CLI configuration intended for the device that originated the configuration pull.

The network administrator must ensure a match between the Cisco Networking Services agent ID as defined on the routing device and the Cisco Networking Services agent ID contained in the directory attribute that corresponds to the configuration intended for the routing device. Within the routing device, the default value of the Cisco Networking Services agent ID is always set to the hostname. If the hostname changes, the Cisco Networking Services agent ID also changes. If the Cisco Networking Services agent ID is set using the CLI, any change will be followed by a message sent to syslog or an event message will be sent.

The Cisco Networking Services agent ID does not address security issues.

Cisco Networking Services Password

The Cisco Networking Services password is used to authenticate the Cisco Networking Services device. You must configure the Cisco Networking Services password the first time a router is deployed, and the Cisco Networking Services password must be the same as the bootstrap password set on the Configuration Engine (CE). If both the router and the CE bootstrap password use their default settings, a newly deployed router will be able to connect to the CE. Once connected, the CE manages the Cisco Networking Services password. Network administrators must ensure not to change the Cisco Networking Services password. If the Cisco Networking Services password is changed, connectivity to the CE will be lost.

Command Scheduler

The Command Scheduler (KRON) Policy for System Startup feature enables support for the Command Scheduler upon system startup.

The Command Scheduler allows customers to schedule fully-qualified EXEC mode CLI commands to run once, at specified intervals, at specified calendar dates and times, or upon system startup. Originally designed to work with Cisco Networking Services commands, Command Scheduler now has a broader application. Using the Cisco Networking Services image agent feature, remote routers residing outside a firewall or using Network Address Translation (NAT) addresses can use Command Scheduler to launch CLI at intervals, to update the image running in the router.

Command Scheduler has two basic processes. A policy list is configured containing lines of fully-qualified EXEC CLI commands to be run at the same time or same interval. One or more policy lists are then scheduled to run after a specified interval of time, at a specified calendar date and time, or upon system startup. Each scheduled occurrence can be set to run either once only or on a recurring basis.

Cisco Networking Services Flow-Through Provisioning

Cisco Networking Services Flow-Through Provisioning provides the infrastructure for automated configuration of large numbers of network devices. Based on Cisco Networking Services event and configuration agents, it eliminates the need for an onsite technician to initialize the device. The result is an automated workflow from initial subscriber-order entry through Cisco manufacturing and shipping to final device provisioning and subscriber billing. This functionality focuses on a root problem of today’s service-provider and other similar business models: use of human labor in activating service.

To achieve such automation, Cisco Networking Services Flow-Through Provisioning relies on standardized configuration templates that you create. However, the use of such templates requires a known fixed hardware configuration, uniform for all subscribers. There is no way to achieve this without manually prestaging each line card or module within each chassis. While the inventory within a chassis is known at

time of manufacture, controlling which line cards or modules are in which slots thereafter is labor-intensive and error-prone.

To overcome these difficulties, Cisco Networking Services Flow-Through Provisioning defines a new set of Cisco IOS commands--the **cns** commands. When a remote router is first powered on, these commands do the following:

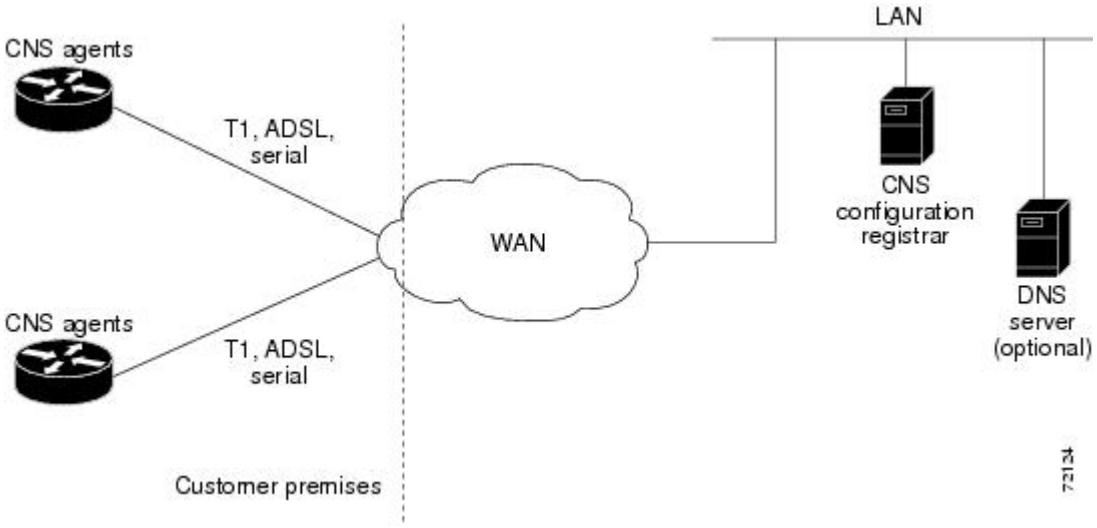
1. To each router interface in turn, applies a preset temporary bootstrap configuration that tries to contact the Cisco Networking Services configuration engine. A successful connection determines the connecting interface.
2. Connects, by way of software called a Cisco Networking Services agent, to a Cisco Networking Services configuration engine housed in a Cisco IE2100 device.
3. Passes to the Cisco Networking Services configuration engine a device-unique ID, along with a human-readable description of the router's line-card or module inventory by product number and location, in XML format.

In turn, the configuration engine does the following:

1. Locates in a Lightweight Directory Access Protocol (LDAP) directory, based on the device IDs, a predefined configuration template for the main chassis and subconfiguration template for each line card or module.
2. Substitutes actual slot numbers from the chassis inventory for the template's slot-number parameters, thus resolving the templates into subscriber-specific configurations that match the true line-card or module slot configuration.
3. Downloads this initial configuration to the target router. The Cisco Networking Services agent directly applies the configuration to the router.

The figure below shows the Cisco Networking Services Flow-Through Provisioning architecture.

Figure 1



Configurations

Cisco Networking Services Flow-Through Provisioning involves three different types of configuration on the remote router:

- Bootstrap configuration

You specify the preset bootstrap configuration on which this solution depends as part of your order from Cisco using Cisco Configuration Express, an existing service integrated with the Cisco.com order-entry tool. You specify a general-subscriber nonspecific bootstrap configuration that provides connectivity to the Cisco Networking Services configuration engine. Cisco then applies this configuration to all the devices of that order in a totally automated manufacturing step. This configuration runs automatically on power-on.

- Initial configuration

The Cisco Networking Services configuration engine downloads an initial configuration, once only, to replace the temporary bootstrap configuration. You can either save or not save it in the router's nonvolatile NVRAM memory:

- ◦ If you save the configuration, the bootstrap configuration is overwritten.
  ◦ If you do not save the configuration, the download procedure repeats each time that the router powers off and then back on. Repeating the download procedure enables the router to update to the current Cisco IOS configuration without intervention.
- Incremental (partial) configuration

On subsequent reboot, incremental or partial configurations are performed to update the configuration without the network having to shut down. Such configurations can be delivered either in a push operation that you initiate or a pull operation on request from the router.

Unique IDs

Key to this solution is the capability to associate, with each device, a simple, manageable, and unique ID that is compatible with your systems for order entry, billing, provisioning, and shipping and can also link your order-entry system to the Cisco order-fulfillment system. Such an ID must have the following characteristics:

- Be available from manufacturing as part of order fulfillment
- Be recordable on the shipping carton and chassis
- Be available to the device's Cisco IOS software
- Be modifiable after the device is first powered up
- Be representative of both a specific chassis and a specific entry point into your network

To define such an ID, Cisco Networking Services Flow-Through Provisioning equips the Cisco Networking Services agent with a new set of commands--the **cns** commands--with which you specify how configurations should be done and, in particular, how the system defines unique IDs. You enable the Cisco IOS software to auto-discover the unique ID according to directions that you specify and information that you provide, such as chassis serial number, MAC address, IP address, and several other possibilities. The **cns** commands are part of the bootstrap configuration of the manufactured device, specified to Cisco Configuration Express at time of order.

Within this scope, Configuration Express and the **cns**commands also allow you to define custom asset tags to your own specifications, which are serialized during manufacture and automatically substituted into the unit's bootstrap configuration.

Cisco appends tags to the carton for all the various types of IDs supported by the **cns**commands, so that these values can be bar-code read at shipping time and fed back into your systems. Alternatively, these IDs are also available through a direct XML-software interface between your system and the Cisco order-status engine, eliminating the need for bar-code reading. The Cisco Networking Services agent also provides a feedback mechanism whereby the remote device can receive XML events or commands to modify the device's ID, in turn causing that same device to broadcast an event indicating the old/new IDs.

Management Point

On most networks, a small percentage of individual remote routers get configured locally. This can potentially be a serious problem, not only causing loss of synchronization across your network but also opening your system to the possibility that an automatic reconfiguration might conflict with an existing configuration and cause a router to become unusable or even to lose contact with the network.

To address this problem, you can designate a management point in your network, typically on the Cisco IE2100 Cisco Networking Services configuration engine, and configure it to keep track of the configurations on all remote routers.

To enable this solution, configure the Cisco Networking Services agent to publish an event on the Cisco Networking Services event bus whenever any change occurs to the running configuration. This event indicates exactly what has changed (old/new), eliminating the need for the management point to perform a highly unscalable set of operations such as telnetting into the device, applying a script, reading back the entire running configuration, and determining the difference between old and new configurations. Additionally, you can arrange for Simple Network Management Protocol (SNMP) notification traps of configuration changes occurring through the SNMP MIB set.

Point-to-Point Event Bus

Today's business environment requires that you be able to ensure your customers a level of service not less than what they are actually paying for. Toward this end, you activate service-assurance applications that broadcast small poll/queries to the entire network while expecting large responses from a typically small subset of devices according to the criteria of the query.

For these queries to be scalable, it is necessary for the replying device to bypass the normal broadcast properties of the event bus and instead reply on a direct point-to-point channel. While all devices need the benefit of the broadcasted poll so that they can all be aware of the query to which they may need to reply, the devices do not have to be aware of each others' replies. Massive copying and retransmission of device query replies, as part of the unnecessary reply broadcast, is a serious scalability restriction.

To address this scalability problem, the Cisco Networking Services event bus has a point-to-point connection feature that communicates directly back to the poller station.

Cisco Networking Services Flow-Through Provisioning provides the following benefits.

Automated Configuration

Cisco Networking Services Flow-Through Provisioning simplifies installation by moving configuration requirements to the Cisco Networking Services configuration engine and allowing the Cisco IOS configuration to update automatically. The registrar uses popular industry standards and technologies such as XML, Active Directory Services Interface (ADSI)/Active Directory, HTTP/Web Server, ATM Switch Processor (ASP), and Publish-Subscribe Event Bus. The Cisco Networking Services configuration agent enables the Cisco Networking Services configuration engine to configure remote routers in a plug-and-play manner.

Unique IP Addresses and Hostname

Cisco Networking Services Flow-Through Provisioning uses DNS reverse lookup to retrieve the hostname by passing the IP address, then assigns the IP address and optionally the hostname to the remote router. Both IP address and hostname are thus guaranteed to be unique.

Reduced Technical Personnel Requirements

Cisco Networking Services Flow-Through Provisioning permits remote routers to be installed by a person with limited or no technical experience. Because configuration occurs automatically on connection to the network, a network engineer or technician is not required for installation.

Rapid Deployment

Because a person with limited or no technical experience can install a remote router immediately without any knowledge or use of Cisco IOS software, the router can be sent directly to its final premises and be brought up without technician deployment.

Direct Shipping

Routers can be shipped directly to the remote end-user site, eliminating warehousing and manual handling. Configuration occurs automatically on connection to the network.

Remote Updates

Cisco Networking Services Flow-Through Provisioning automatically handles configuration updates, service additions, and deletions. The Cisco Networking Services configuration engine performs a push operation to send the information to the remote router.

Security

Event traffic to and from the remote router is opaque to unauthorized listeners or intruders to your network. Cisco Networking Services agents leverage the latest security features in Cisco IOS software.

Cisco Networking Services Zero Touch

The Cisco Networking Services Zero Touch feature provides a zero touch deployment solution where the router contacts a Cisco Networking Services configuration engine to retrieve its full configuration automatically. This capability is made possible through a single generic bootstrap configuration file common across all service provider end customers subscribing to the services. Within the Cisco Networking Services framework, customers can create this generic bootstrap configuration without device-specific or network-specific information such as interface type, line type, or controller type (if applicable).

The Cisco Networking Services connect functionality is configured with a set of Cisco Networking Services connect templates. A Cisco Networking Services connect profile is created for connecting to the Cisco Networking Services configuration engine and to implement the Cisco Networking Services connect templates on a Customer Premise Equipment (CPE) router. Cisco Networking Services connect variables can be used as placeholders within a Cisco Networking Services connect template configuration. These variables, such as the active DLCI, are substituted with real values before the Cisco Networking Services connect templates are sent to the router's parser.

To use the zero touch functionality, the router that is to be initialized must have a generic bootstrap configuration. This configuration includes Cisco Networking Services connect templates, Cisco Networking Services connect profiles, and the **cns config initial** command. This command initiates the Cisco Networking Services connect function.

The Cisco Networking Services connect functionality performs multiple ping iterations through the router's interfaces and lines, as well as any available controllers. For each iteration, the Cisco Networking Services connect function attempts to ping the Cisco Networking Services configuration engine. If the ping is successful, the pertinent configuration information can be downloaded from the Cisco Networking Services configuration engine. If connectivity to the Cisco Networking Services configuration engine is unsuccessful, the Cisco Networking Services connect function removes the configuration applied to the

selected interface, and the Cisco Networking Services connect process restarts with the next available interface specified by the Cisco Networking Services connect profile.

The Cisco Networking Services Zero Touch feature provides the following benefits:

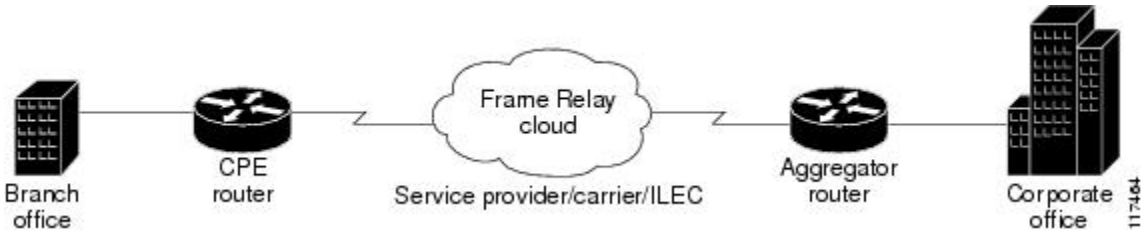
- Ensures consistent Cisco Networking Services commands between Cisco IOS Release 12.3 and 12.3T.
- Use of a channel service unit (E1 or T1 controller) is allowed.

Cisco Networking Services Frame Relay Zero Touch

The Cisco Networking Services Frame Relay Zero Touch feature provides a Cisco Networking Services zero touch deployment solution over Frame Relay where the CPE router discovers its data-link connection identifier (DLCI) and IP address dynamically, and then contacts a Cisco Networking Services engine to retrieve its full configuration automatically. This capability is made possible through a single generic bootstrap configuration file common across all service provider end customers subscribing to the services. Within the Cisco Networking Services framework, customers who deploy Frame Relay can create this generic bootstrap configuration without device-specific or network-specific information such as the DLCI, IP address, interface type, controller type (if applicable), or the next hop interface used for the static default route.

The following image illustrates a typical customer network architecture using Frame Relay.

Figure 2



The CPE router is deployed at multiple sites. Each site connects to a Frame Relay cloud through a point-to-point permanent virtual circuit (PVC). Connectivity from the Frame Relay cloud to the corporate office is through a PVC that terminates at the corporate office. IP traffic sent to the Cisco Networking Services configuration engine is routed through the corporate office. The PVC is identified by its DLCI. The DLCI can vary between branch offices. In order to support zero touch deployment, the CPE router must be able to learn which DLCI to use to connect to the Cisco Networking Services configuration engine.

To support the zero touch capability, the Frame Relay functionality has been modified in the following two ways:

- A new Cisco IOS command, the **ip address dynamic**command has been introduced to discover the CPE router's IP address dynamically based on the aggregator router's IP address. To configure IP over Frame Relay, the local IP address must be configured on the interface.
- The CPE router can now read Local Management Interface (LMI) messages from a Frame Relay switch and determine the list of available DLCIs.

The Cisco Networking Services connect functionality is configured with a set of Cisco Networking Services connect templates. A Cisco Networking Services connect profile is created for connecting to the Cisco Networking Services configuration engine and to implement the Cisco Networking Services connect templates on a CPE router. Cisco Networking Services connect variables can be used as placeholders within a Cisco Networking Services connect template configuration. These variables, such as the active

DLCI, are substituted with real values before the Cisco Networking Services connect templates are sent to the router's parser.

When a CPE router is placed in a Frame Relay network, it contains a generic bootstrap configuration. This configuration includes customer-specific Frame Relay configuration (including the LMI type), Cisco Networking Services connect templates, Cisco Networking Services connect profiles, and the **cns config initial** command. This command initiates the Cisco Networking Services connect function.

The Cisco Networking Services connect functionality begins by selecting the first available controller or interface specified by the Cisco Networking Services connect profile and then performs multiple ping iterations through all the associated active DLCIs. For each iteration, the Cisco Networking Services connect function attempts to ping the Cisco Networking Services configuration engine. If the ping is successful, the pertinent configuration information can be downloaded from the Cisco Networking Services configuration engine.

When iterating over the active DLCIs on a Frame Relay interface, the router must be able to automatically go through a list of active DLCIs returned by the LMI messages for that interface and select an active DLCI to use. When more than one of the active DLCIs allow IP connectivity to the Cisco Networking Services configuration engine, the DLCI used will be the first one tried by the Cisco Networking Services connect functionality. If the ping attempt is unsuccessful, the next active DLCI is tried and so on. If connectivity to the Cisco Networking Services configuration engine is unsuccessful for all active DLCIs, the Cisco Networking Services connect function removes the configuration applied to the selected controller or interface, and the Cisco Networking Services connect process restarts with the next available controller or interface specified by the Cisco Networking Services connect profile.

The Cisco Networking Services Frame Relay Zero Touch feature provides the following benefits:

- A service provider can have a single common bootstrap configuration.
- The generic bootstrap configuration does not require the IP address to be hard-wired.
- The point-to-point DLCI does not need to be known in advance.
- IP directly over Frame Relay is allowed.
- Use of a channel service unit (E1 or T1 controller) is allowed.

Zero Touch Deployment

The Cisco Zero Touch deployment (ZTD) solution enables the router to retrieve configuration files from the remote DHCP server during the initial router deployment. You need a bootstrap configuration to communicate between the router and the remote server. The bootstrap configuration provides specific information about a device. This bootstrap configuration can be pre-installed on the device or can be retrieved from the DHCP serv er. Another method of retrieving the bootstrap configuration information, using the DHCP Option 43, is introduced in Cisco IOS Release 15.1(1)T. To accommodate situations where routers cannot have a pre-installed bootstrap configuration, a deployment model which uses DHCP Option 43 messages is used. Cisco recommends the usage of DHCP Option 43 message based on RFC 2132. You can use the DHCP Option 43 message to provide vendor-specific information in the form of ASCII codes to the DHCP server.

The DHCP Option 43 message supplies the necessary information that is normally provided in the bootstrap configuration to the DHCP client. When the DHCP client issues a DHCP IP address request to the DHCP server, the DHCP server sends out the IP address and a DHCP Option 43 message, if the DHCP Option 43 message is pre-configured on the DHCP server. Within this DHCP Option 43 message, pre-defined parameterized Cisco Networking Services commands are provided to the DHCP client. The DHCP client receives the DHCP Option 43 message and then forwards it to the Cisco Networking Services DHCP Option 43 message processing unit for further processing. A timer for three minutes is set. The initial configuration file that is set to download is checked after five minutes. If the file download is succesful the process is complete. If the file download fails, check if the Cisco Networking Services DHCP Option 43

message generated is correct and fix it if there is problem. Power cycle the router to retry the Cisco Networking Services DHCP Option 43 message processing.

At router system initiation time, there are following two ways to initiate the DHCP IP address request to enable the DHCP Option 43 message to be sent to the router:

1. If the router is enabled with startup configuration, ZTD can be enabled by using the **ip address dhcp** and the **cns dhcp** configuration commands.
2. If the router is not enabled with startup configuration, the Autoinstall feature automatically initializes the ip address dhcp configuration command, which enables the ZTD. For more information about the Autoinstall feature, see the Overview - Basic Configuration of a Cisco Networking Device module in the *Cisco IOS Configuration Fundamentals Configuration Guide* .

• Cisco Networking Services Parameterized Commands Defined Within DHCP Option 43 Message to Enable ZTD, page 17

Cisco Networking Services Parameterized Commands Defined Within DHCP Option 43 Message to Enable ZTD

The values configured using the **cns config initial**, **cns config partial**, **cns config id**, **cns event**, **cns exec**, and **cns trusted-server all-agents** commands are used as parameters to construct the DHCP Option 43 message to enable ZTD. The DHCP Option 43 message provides these pre-defined parameterized commands to the DHCP client, which enables the client to decode and read the messages sent by the DHCP Server.

• Constructing a DHCP Option 43 message, page 17

Constructing a DHCP Option 43 message

The DHCP Option 43 message is presented in the type/value (TV) format. The DHCP Option 43 is used by clients and servers to exchange vendor- specific information. When you use the vendor-specific option (Option 43), you must specify the data using hexadecimal ASCII values. For more information on the option command refer to Cisco IOS IP Addressing Command Reference Guide.

Note The maximum DHCP Option 43 size is 2500 bytes.

Following are the parameters used by the Cisco Networking Services to construct the DHCP Option 43 message to enable ZTD:

```
<DHCP-typecode><feature-opcode><version><debug-option>;<arglist>
```

The following table describes the parameters and their syntax.

Table 2 Parameters of DHCP Option 43 Message

Parameter	Description
DHCP-typecode	Specifies the DHCP suboption type. The DHCP suboption type for Cisco Networking Services is 3.

Parameter	Description
feature-opcode	There are two types of feature op-codes--Active (A) and Passive (P). The feature op-codes for Cisco Networking Service are A and P templates.
Active Template	This code connects to the CE and sends a request for a configuration. If the CE is not reached, the router trys to download a configuration from the CE until the configuration is downloaded.
Passive Template	This code connects to the event gateway and then waits for the configuration.
version	Indicates the version of template to be used by Cisco Networking Service.
debug-option	Indicates if debug messages have to be generated during the processing of the DHCP Option 43 messages. Debug OFF is recommended for normal processing and debug ON can be used for debugging the processing of DHCP Option 43 message. The following are the two debug options: • D--debug option is ON • N--debug option is OFF
;	Delimiter used to separate the parameters.
arglist	List of named arguments for the command, separated by semi-colon. To use the default value for an argument, you need not specify values for that parameter. Include a parameter and its value only when its default value does not serve the need. Letter codes are used to identify the arguments. Name and value pairs can be listed in any order and are delimited by a semi-colon.

The following table lists the arguments for configuring the Cisco Networking Service ID and the initiator profile parameters used for configuring the Cisco Networking Service Active Template configuration agent.

Table 3 Argument Lists for Cisco Networking Service Active Template (Cisco Networking Service Indicators)

Parameter	Letter Code	Values	Parameter to CLI Mapping
Sample Letter Code	Sample CLI Mapping		

Parameter	Letter Code	Values	Parameter to CLI Mapping	
cns ID	A	(Optional) Indicates the Cisco Networking Service ID. The default is hostname. 1- Indicates a custom string to be used. 2- Indicates the MAC-address of the interface used. 3- Indicates the hardware serial number to be used. 4- Indicates Unified Display Interface (UDI).	`A1881-ap A4`	`Router(config)# cns id string 881-ap event` `Router(config)# cns id string 881-ap`
CE Address	B	(Required) Specifies the IPv4/IPv6 address/ hostname. If using hostname, set the DNS-server option for DHCP.	`B10.10.10.1`	`Router(config)# cns config initial 10.10.10.1`
CE config server port	C	(Optional) Specifies the CE config server port numeric string values between 0 and 65535. The default value is 80.	`C11025`	`Router(config)# cns config initial ce-address 11025`
Source interface	D	(Optional) Indicates the source interface name.	`DF0/1`	`Router(config)# cns config initial ce-address source fastethernet 0/1`
Status Destination	E	(Optional) Indicates the destination status. The default value is syslog. 1- <URL> -http, should be followed by the URL. The default value is None.	`F/cns/config.asp`	`Router(config)# cns config initial ce-address page /cns/ config.asp`

Parameter	Letter Code	Values	Parameter to CLI Mapping	
Config no-persist	I	(Optional) Specifies the configuration conditions to NVRAM 1-no-persist: Do not write configuration to NVRAM 1-persist: Write configuration to NVRAM Default- persist	`I1`	`Router(config)# cns config initial no-persist`

The following table lists the arguments for configuring the Cisco Networking Service ID and the initiator profile parameters used for configuring the Cisco Networking Service Passive Template configuration agent.

Table 4 Argument Lists for Cisco Networking Service Active Template (Cisco Networking Service Indicators)

Parameter	Letter Code	Values	Parameter to CLI Mapping	
Sample Letter Code	Sample CLI Mapping			
cns ID	A	(Optional) Indicates the Cisco Networking Service ID. The default is hostname. 1- Indicates a custom string to be used. 2- Indicates the MAC-address of the interface used. 3- Indicates the hardware serial number to be used. 4- Indicates Unified Display Interface (UDI).	`A1881-ap` `A4`	`Router(config)# cns id string 881-ap event` `Router(config)# cns id string 881-ap`
CE Address	B	(Required) Specifies the IPv4/IPv6 address/ hostname. If using hostname, set the DNS-server option for DHCP.	`B10.10.10.1`	`Router(config)# cns config initial 10.10.10.1`

Parameter	Letter Code	Values	Parameter to CLI Mapping	
CE config server port	C	(Optional) Specifies the numeric string values between 0 and 65535. The default value is 80.	C11025	Router(config)# **cns config initial ce-address 11025**
Source interface	D	(Optional) Indicates the source interface name.	DF0/1	Router(config)# **cns config initial ce-address source fastethernet 0/1**
CE event gateway port	G	(Optional) Specifies CE event gateway port numeric string values between 0 and 65535. The default value is 11011.	G11025	Router(config)# **cns event ce-address 11025**

Examples of Letter Code Mappings for Active Template

Example 1

In this example, in response to a DHCP IP address request sent by the DHCP client, the DHCP server sends an Option 43 message such as **3P2N;B10.10.10.1** to the DHCP client. The DHCP client forwards the Option 43 message to the Cisco Networking Service. The Cisco Networking Service verifies if the Option 43 message is allowed to process. Option 43 messages are allowed to process by the Cisco Networking Service if the cns dhcp command is enabled on the Cisco Networking Service.

The ASCII data shown in this Option 43 message consists of types and values as shown in the following table.

Table 5 Types and Values for Sample Option 43 Command

Type	Value
3	P2N;B10.10.10.1

This message is decoded into tokens using the above arguments list. The parameters mapped for the 3P2N;B10.10.10.1 message using the arguments list are as follows:

P--Active template code

2--Version number of the Active template

N--Debug option which is OFF

;--Delimiter before the arglist

B10.10.10.1--CE address parameter name value pair

The Cisco Networking Service constructs the following commands and sends to the remote management server to request the initial configuration file. A timer is set for five minutes.

```
Router(config)# cns event 10.10.10.1
Router(config)# cns config partial 10.10.10.1 inventory
Router(config)# cns exec
Router(config)# cns trusted-server all-agents 10.10.10.1
```

The initial configuration file that is downloaded is checked. If the file download is successful, the process is complete.

Example 2

In this example, in response to a DHCP IP address request sent by the DHCP client, the DHCP server sends an Option 43 message such as **3P1N;A1881-ap;B10.10.10.1;J11024** to the DHCP client. The DHCP client forwards the Option 43 message to the Cisco Networking Service. The Cisco Networking Service verifies if the Option 43 message is allowed to process. Option 43 messages are allowed to process by the Cisco Networking Service if the **cns dhcp** command is enabled on the Cisco Networking Service.

The ASCII data shown in this Option 43 message consists of types and values shown in the following table.

Table 6 Types and Values for Sample Option 43 Command

Type	Value
3	P1N;A1881-ap;B10.10.10.1;J11024

This message is decoded into tokens using the above arguments list. The parameters mapped for the 3P1N;A1881-ap;B10.10.10.1;C11024 message using the arguments list are as follows:

P--Active template code

1--Version number of the Active template

N--Debug option which is OFF

;--Delimiter before the arglist

881-ap-Active template string values

B10.10.10.1--CE address parameter name value pair

J11024--Config server port value

How to Configure Cisco Networking Services

Deploying the Cisco Networking Services Router

Perform this task to manually install an initial Cisco Networking Services configuration.

Your remote router arrives from the factory with a bootstrap configuration. Upon initial power-on, the router automatically pulls a full initial configuration from the Cisco Networking Services configuration engine, although you can optionally arrange for this manually as well. After initial configuration, you can optionally arrange for periodic incremental (partial) configurations for synchronization purposes.

- Initial Cisco Networking Services Configuration, page 23
- Incremental Configuration, page 23

Initial Cisco Networking Services Configuration

Initial configuration of the remote router occurs automatically when the router is initialized on the network. Optionally, you can perform this configuration manually.

Cisco Networking Services assigns the remote router a unique IP address or hostname. After resolving the IP address (using Serial Line Address Resolution Protocol (SLARP), ATM Inverse ARP (ATM InARP), or PPP protocols), the system optionally uses Domain Name System (DNS) reverse lookup to assign a hostname to the router and invokes the Cisco Networking Services agent to download the initial configuration from the Cisco Networking Services configuration engine.

Incremental Configuration

Incremental or partial configuration allows the remote router to be incrementally configured after its initial configuration. You must perform these configurations manually through the Cisco Networking Services configuration engine. The registrar allows you to change the configuration templates, edit parameters, and submit the new configuration to the router without a software or hardware restart.

Before you can configure an incremental configuration, Cisco Networking Services must be operational and the required Cisco Networking Services agents configured.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns template connect** *name*
4. **cli** *config-text*
5. Repeat Step 4 to add all required CLI commands.
6. **exit**
7. **cns connect** *name* [**retry-interval** *interval-seconds*] [**retries** *number-retries*] [**timeout** *timeout-seconds*] [**sleep** *sleep-seconds*]
8. Do one of the following:
   - **discover** {**line** *line-type* | **controller** *controller-type* | **interface** [*interface-type*]}
   -
   - **template** *name*
9. **exit**
10. **cns config initial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory**]
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns template connect** *name* **Example:** `Router(config)# cns template connect template 1`	Enters Cisco Networking Services template connect configuration mode and defines the name of a Cisco Networking Services connect template.

	Command or Action	Purpose
Step 4	**cli** *config-text* **Example:** `Router(config-templ-conn)# cli encapsulation ppp`	Specifies commands to configure the interface.
Step 5	Repeat Step 4 to add all required CLI commands. **Example:** `Router(config-templ-conn)# cli ip directed-broadcast`	Repeat Step 4 to add other CLI commands to configure the interface or to configure the modem lines.
Step 6	**exit** **Example:** `Router(config-templ-conn)# exit`	Exits Cisco Networking Services template connect configuration mode and completes the configuration of a Cisco Networking Services connect template. **Note** Entering the **exit** command is required. This requirement was implemented to prevent accidentally entering a command without the **cli** command.
Step 7	**cns connect** *name* [**retry-interval** *interval-seconds*] [**retries** *number-retries*] [**timeout** *timeout-seconds*] [**sleep** *sleep-seconds*] **Example:** `Router(config)# cns connect profile-1 retry-interval 15 timeout 90`	Enters Cisco Networking Services connect configuration mode and defines the parameters of a Cisco Networking Services connect profile for connecting to the Cisco Networking Services configuration engine.

	Command or Action	Purpose

Step 8	Do one of the following:
	• **discover** {**line** *line-type* | **controller** *controller-type* | **interface** [*interface-type*]}
	•
	• **template** *name*
	
	Example:
	`Router(config-cns-conn)# discover interface serial`
	
	Example:
	
	Example:
	`Router(config-cns-conn)# template template-1`
		(Optional) Configures a generic bootstrap configuration.
		• **discover** --Defines the interface parameters within a Cisco Networking Services connect profile for connecting to the Cisco Networking Services configuration engine.
		or
		• **template** --Specifies a list of Cisco Networking Services connect templates within a Cisco Networking Services connect profile to be applied to a router's configuration.

Step 9	**exit** **Example:** `Router(config-cns-conn)# exit`	Exits Cisco Networking Services connect configuration mode and returns to global configuration mode.
Step 10	**cns config initial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory**] **Example:** `Router(config)# cns config initial 10.1.1.1 no-persist`	Starts the Cisco Networking Services configuration agent, connects to the Cisco Networking Services configuration engine, and initiates an initial configuration. You can use this command only before the system boots for the first time. **Note** The optional **encrypt** keyword is available only in images that support Secure Socket Layer (SSL). **Caution** If you write the new configuration to NVRAM by omitting the **no-persist**keyword, the original bootstrap configuration is overwritten.
Step 11	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring the Cisco Networking Services Event and EXEC Agents

Perform this task to enable and configure the Cisco Networking Services Event and EXEC agents.

Cisco Networking Services Event Agent Parameters

The Cisco Networking Services event agent command--**cns event**--has several parameters that can be configured. The **failover-time** keyword is useful if you have a backup Cisco Networking Services event gateway configured. If the Cisco Networking Services event agent is trying to connect to the gateway and it discovers that the route to the backup gateway is available before the route to the primary gateway, the *seconds* argument specifies how long the Cisco Networking Services event agent will continue to search for a route to the primary gateway before attempting to link to the backup gateway.

Unless you are using a bandwidth-constrained link, you should set a keepalive timeout and retry count. Doing so allows the management network to recover gracefully should a Cisco IE2100 configuration engine ever fail. Without the keepalive data, such a failure requires manual intervention on every device. The *seconds* value multiplied by the *retry-count* value determines the length of idle time before the Cisco Networking Services event agent will disconnect and attempt to reconnect to the gateway. We recommend a minimum *retry-count* value of 2.

If the optional **source** keyword is used, the source IP address might be a secondary IP address of a specific interface to allow a management network to run on top of a production network.

Note Although other Cisco Networking Services agents may be configured, no other Cisco Networking Services agents are operational until the **cns event** command is entered because the Cisco Networking Services event agent provides a transport connection to the Cisco Networking Services event bus for all other Cisco Networking Services agents.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns config partial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**source** *interface name*] [**inventory**]
4. **logging cns-events** [*severity-level*]
5. **cns exec** [*host-name* | *ip-address*] [**encrypt**[enc-*port-number*]] [*port-number*] [**source** *ip-address*]
6. **cns event** {*hostname* | *ip-address*} [**encrypt**] [*port-number*] [**backup**] [**failover-time** *seconds*] [**keepalive** *seconds retry-count*] [**source** *ip-address*][**clock-timeout** *time*] [**reconnect** *time*]
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns config partial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**source** *interface name*] [**inventory**] **Example:** `Router(config)# cns config partial 172.28.129.22 80`	(Optional) Starts the Cisco Networking Services configuration agent, which provides Cisco Networking Services configuration services to Cisco IOS clients, and initiates an incremental (partial) configuration. • Use the optional *port-number* argument to specify the port number for the configuration server. The default is 80. • Use the optional **source** keyword and *ip-address* argument to specify the use of an IP address as the source for Cisco Networking Services configuration agent communications. • Use the optional **inventory** keyword to send an inventory of the line cards and modules in the router to the Cisco Networking Services configuration engine as part of the HTTP request. **Note** The optional **encrypt** keyword is available only in images that support SSL.
Step 4	**logging cns-events** [*severity-level*] **Example:** `Router(config)# logging cns-events 2`	(Optional) Enables XML-formatted system event message logging to be sent through the Cisco Networking Services event bus. • Use the optional *severity-level* argument to specify the number or name of the desired severity level at which messages should be logged. The default is level 7 (debugging).
Step 5	**cns exec** [*host-name* | *ip-address*] [**encrypt**[enc-*port-number*]] [*port-number*] [**source** *ip-address*] **Example:** `Router(config)# cns exec 10.1.2.3 93 source 172.17.2.2`	(Optional) Enables and configures the Cisco Networking Services EXEC agent, which provides Cisco Networking Services EXEC services to Cisco IOS clients. • Use the optional *port-number* argument to specify the port number for the EXEC server. The default is 80. • Use the optional **source** keyword and *ip-address* argument to specify the use of an IP address as the source for Cisco Networking Services EXEC agent communications. **Note** The optional **encrypt keyword** is available only in images that support SSL.

	Command or Action	Purpose
Step 6	**cns event** {*hostname* | *ip-address*} [**encrypt**] [*port-number*] [**backup**] [**failover-time** *seconds*] [**keepalive** *seconds retry-count*] [**source** *ip-address*][**clock-timeout** *time*] [**reconnect** *time*] **Example:** `Router(config)# cns event 172.28.129.22 source 172.22.2.1`	Configures the Cisco Networking Services event gateway, which provides Cisco Networking Services event services to Cisco IOS clients. • The optional **encrypt**keyword is available only in images that support SSL. • Use the optional *port-number* argument to specify the port number for the event server. The default is 11011 with no encryption and 11012 with encryption. • Use the optional **backup** keyword to indicate that this is the backup gateway. Before configuring a backup gateway, ensure that a primary gateway is configured. • Use the optional **failover-time** keyword and *seconds* argument to specify a time interval in seconds to wait for the primary gateway route after the route to the backup gateway is established. • Use the optional **keepalive** keyword with the *seconds* and *retry-count* arguments to specify the keepalive timeout in seconds and the retry count. • Use the optional **source** keyword and *ip-address* argument to specify the use of an IP address as the source for Cisco Networking Services event agent communications. • Use the optional **clock-timeout** keyword to specify the maximum time, in minutes, that the Cisco Networking Services event agent will wait for the clock to be set for transports (such as SSL) that require an accurate clock. • Use the optional **reconnect** keyword to specify the configurable upper limit of the maximum retry timeout. **Note** Until the **cns event** command is entered, no transport connections to the Cisco Networking Services event bus are made and therefore no other Cisco Networking Services agents are operational.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Troubleshooting Tips

- Use the **show cns event connections**command to check that the Cisco Networking Services event agent is connected to the Cisco Networking Services event gateway.
- Use the **show cns event subject**command to check that the image agent subject names are registered. Subject names for the Cisco Networking Services image agent begin with cisco.mgmt.cns.image.

Enabling Cisco Networking Service to Receive DHCP Option 43 Message

Perform this task to enable a Cisco Networking Service with permission to process the incoming DHCP Option 43 message.

Cisco IOS Subsystem

Ensure that the following Cisco IOS subsystems are supported:

- DHCP client
- Cisco Networking Service

Software Requirements

- SSH client
- HTTP(S) 1.1 listener
- HTTP(S) 1.1 client
- SOAP
- XML parser
- Cisco Networking Service agent libraries

External Devices

- Configuration Engine
- DHCP server with Option 43 message supported

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns dhcp**
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns dhcp** **Example:** `Router(config)# cns dhcp`	Enables Cisco Networking Service with permission to process the incoming DHCP Option 43 message.

	Command or Action	Purpose
Step 4	**exit** **Example:** `Router# exit`	Exits global configuration mode.

Configuring the Cisco Networking Services Image Agent

Perform this task to configure Cisco Networking Services image agent parameters using CLI commands.

- Cisco Networking Services Image Agent ID, page 31
- What to Do Next, page 33

Cisco Networking Services Image Agent ID

Cisco Networking Services uses a unique identifier to identify an image agent associated with that Cisco IOS device. Using the same process as Cisco Networking Services event and configuration agents, the configuration of the **cns id** command determines whether an IP address or MAC address of a specified interface, the hardware serial hardware number of the device, an arbitrary text string, or the hostname of the device is used as the image ID. By default, the system uses the hostname of the device.

The Cisco Networking Services image ID is sent in the content of the messages sent by the image agent and allows an application to know the unique image ID of the Cisco IOS device that generated the message. A password can be configured and associated with the image ID in the image agent messages.

- To configure the Cisco Networking Services image agent to use HTTP or HTTP over SSL (HTTPS) to communicate with an image server, you need to know the URL for the image server and the URL to which status messages can be sent.
- If you are using HTTPS to communicate with the image server, you must set up security certificates to allow the server to be authenticated by the image agent when the connection is established.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Do one of the following:
   - **cns id** *type number* {**ipaddress**| **mac-address**} [**event**| **image**]
   -
   - **cns id** {**hardware-serial**| **hostname**| **string** *text*} [**event**| **image**]
4. **cns password** *password*
5. **cns image** [**server** *server-url*[**status** *status-url*]]
6. **cns image password** *image-password*
7. **cns image retry** *seconds*
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	Do one of the following:
	• **cns id** *type number* {**ipaddress**| **mac-address**} [**event**| **image**]
	•
	• **cns id** {**hardware-serial**| **hostname**| **string** *text*} [**event**| **image**]
	
	Example:
	`Router(config)# cns id fastethernet 0/1 ipaddress image`
	
	Example:
	
	Example:
	`Router(config)# cns id hardware-serial image`
		Specifies a unique Cisco Networking Services ID and interface type and number from which to retrieve the unique ID.
		or
		Specifies a unique Cisco Networking Services ID assigned from the hardware serial number, device hostname, or an arbitrary text string.
		The following information applies to either version of the syntax.
		• Use the **event** keyword to specify an event agent ID.
		• Use the **image** keyword to specify an image agent ID.
		• If no keywords are used, the configuration agent ID is configured.

Step 4	**cns password** *password* **Example:** `Router(config)# cns password password1`	Specifies a password for the Cisco Networking Services ID. You must configure the Cisco Networking Services password the first time a router is deployed, and the Cisco Networking Services password must be the same as the bootstrap password set on the Configuration Engine (CE).

	Command or Action	Purpose
Step 5	**cns image** [**server** *server-url*[**status** *status-url*]] **Example:** `Router(config)# cns image server https://10.21.2.3/cns/imgsvr status https://10.21.2.3/cns/status/`	Enables Cisco Networking Services image agent services and specifies the URL of the image distribution server. • Use the optional **status** keyword and *status-url* argument to specify the URL of a web server to which error messages are written. • If the **status** keyword and *status-url* argument are not specified, status messages are sent as events on the Cisco Networking Services Event Bus. To view the status messages on the Cisco Networking Services Event Bus, the Cisco Networking Services event agent must be configured.
Step 6	**cns image password** *image-password* **Example:** `Router(config)# cns image password abctext`	(Optional) Specifies a password for Cisco Networking Services image agent services. • If a password is configured, the password is included with the image ID in Cisco Networking Services image agent messages sent out by the image agent. The receiver of these messages can use this information to authenticate the sending device.
Step 7	**cns image retry** *seconds* **Example:** `Router(config)# cns image retry 240`	(Optional) Specifies an image upgrade retry interval in seconds. • The default interval is 60 seconds.
Step 8	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns the router to privileged EXEC mode.

What to Do Next

Proceed to the Retrieving a Cisco Networking Services Image from a Server, page 35 section to connect to the web server and download an image.

If any of the commands in the task fail, proceed to the Troubleshooting Cisco Networking Services Agents, page 42 section to try to determine the problem.

Configuring Cisco Networking Services Security Features

Perform this task to configure Cisco Networking Services security features.

- Cisco Networking Services Trusted Servers, page 33

Cisco Networking Services Trusted Servers

Use the **cns trusted-server** command to specify a trusted server for an individual Cisco Networking Services agent or for all the Cisco Networking Services agents. To avoid security violations, you can build

a list of trusted servers from which Cisco Networking Services agents can receive messages. An attempt to connect to a server not on the list will result in an error message being displayed.

Configure a Cisco Networking Services trusted server when a Cisco Networking Services agent will redirect its response to a server address that is not explicitly configured on the command line for the specific Cisco Networking Services agent. For example, the Cisco Networking Services EXEC agent may have one server configured but receive a message from the Cisco Networking Services event bus that overrides the configured server. The new server address has not been explicitly configured, so the new server address is not a trusted server. An error will be generated when the Cisco Networking Services exec agent tries to respond to this new server address unless the **cns trusted-server** command has been configured for the new server address.

Cisco Networking Services Security Enhancement

Cisco Networking Services messages can be configured to use the Cisco Networking Services SOAP message structure, in which the username and password are authenticated. If AAA is configured, then Cisco Networking Services SOAP messages will be authenticated with AAA. If AAA is not configured, there will be no authentication.

Use the **cns aaa authentication**command to determine whether the Cisco Networking Services messages are using AAA security or not. If the **cns aaa authentication**command is configured, then all incoming SOAP messages into the device are authenticated by AAA.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns trusted-server** {**all-agents** | **config** | **event** | **exec** | **image**} *name*
4. **cns message format notification** [**version 1**| **version 2**]
5. **cns aaa authentication** *authentication-method*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**cns trusted-server** {**all-agents** | **config** | **event** | **exec** | **image**} *name* **Example:** `Router(config)# cns trusted-server event 10.19.2.5`	Configures a Cisco Networking Services trusted server for the specified hostname or IP address.
Step 4	**cns message format notification** [**version 1**| **version 2**] **Example:** `Router(config)# cns message format notification version 1`	Configures the message format for notification messages from a Cisco Networking Services device. Received messages which do not conform to the configured message format are rejected. Use version 1 to configure the non-SOAP message format. Use version 2 for SOAP message format.
Step 5	**cns aaa authentication** *authentication-method* **Example:** `Router(config)# cns aaa authentication method1`	Enables Cisco Networking Services AAA options. **Note** The authentication methods must be configured within AAA.

Retrieving a Cisco Networking Services Image from a Server

Perform this task to poll the image distribution server using HTTP or HTTPS.

This task assumes that you have already configured the Cisco Networking Services image agent using the tasks in the Configuring the Cisco Networking Services Image Agent, page 31 section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns image retrieve** [**server** *server-url*[**status** *status-url*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns image retrieve** [**server** *server-url*[**status** *status-url*]] **Example:** `Router(config)# cns image retrieve server https://10.19.2.3/imgsvr/ status https://10.19.2.3/imgsvr/ status/`	Contacts a Cisco Cisco Networking Services image distribution server and downloads a new image if a new image exists. • Use the optional **status** keyword and *status-url* argument to specify the URL of a web server to which status messages are written. • If the **server** and **status** keywords are not specified, the server and status URLs configured with the **cns image** command are used. **Note** We recommend using the **cns trusted-server** command to specify the host part of the server or status URL as a trusted server.

• Troubleshooting Tips, page 36

Troubleshooting Tips

- If the web server appears to be down, use the **ping** command to check connectivity.
- If using HTTP, use the **show ip http client all** command to display information about HTTP clients and connections.

Retrieving a Cisco Networking Services Configuration from a Server

Use this task to request the configuration of a device from a configuration server. Use the **cns trusted-server** command to specify which configuration server can be used (trusted).

This task assumes that you have specified a trusted server using tasks in the Cisco Networking Services Security Enhancement, page 9 section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns config retrieve** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**overwrite-startup**] [**retry** *retries* **interval** *seconds*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns config retrieve** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**overwrite-startup**] [**retry** *retries* **interval** *seconds*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory** **Example:** `Router(config)# cns config retrieve server1 retry 5 interval 45`	Allows the router to retrieve configuration data from a web server. • The **retry** keyword is a number in the range 1 to 100, and will prompt for an **interval** in the range 1 to 3600 seconds.

• Troubleshooting Tips, page 37

Troubleshooting Tips

If you need to stop the retrieval process, enter the Ctrl+Shift+6 key sequence.

Configuring Command Scheduler Policy Lists and Occurrences

Perform this task to set up Command Scheduler policy lists of EXEC Cisco Networking Services commands and configure a Command Scheduler occurrence to specify the time or interval after which the Cisco Networking Services commands will run.

Command Scheduler Policy Lists

Policy lists consist of one or more lines of fully-qualified EXEC CLI commands. All commands in a policy list are executed when the policy list is run by Command Scheduler using the **kron occurrence** command. Use separate policy lists for CLI commands that are run at different times. No editor function is available, and the policy list is run in the order in which it was configured. To delete an entry, use the **no** form of the

cli command followed by the appropriate EXEC command. If an existing policy list name is used, new entries are added to the end of the policy list. To view entries in a policy list, use the **show running-config** command. If a policy list is scheduled to run only once, it will not be displayed by the **show running-config** command after it has run.

Policy lists can be configured after the policy list has been scheduled, but each policy list must be configured before it is scheduled to run.

Command Scheduler Occurrences

An occurrence for Command Scheduler is defined as a scheduled event. Policy lists are configured to run after a specified interval of time, at a specified calendar date and time, or upon system startup. Policy lists can be run once, as a one-time event, or as recurring events over time.

Command Scheduler occurrences can be scheduled before the associated policy list has been configured, but a warning will advise you to configure the policy list before it is scheduled to run.

The clock time must be set on the routing device before a Command Scheduler occurrence is scheduled to run. If the clock time is not set, a warning message will appear on the console screen after the **kron occurrence** command has been entered. Use the **clock** command or Network Time Protocol (NTP) to set the clock time.

The EXEC CLI to be run by Command Scheduler must be tested on the routing device to determine if it will run without generating a prompt or allowing execution interruption by keystrokes. Initial testing is important because Command Scheduler will delete the entire policy list if any CLI syntax fails. Removing the policy list ensures that any CLI dependencies will not generate more errors.

If you use the **conditional** keyword with the **kron policy-list** command, execution of the commands will stop when an error is encountered.

Note

- No more than 31 policy lists can be scheduled to run at the same time.
- If a one-time occurrence is scheduled, the occurrence will not be displayed by the **show running-config** command after the occurrence has run.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **kron policy-list** *list-name* [**conditional**]
4. **cli** *command*
5. **exit**
6. **kron occurrence** *occurrence-name* [**user** *username*] {**in**[[*numdays:*]*numhours:*]*nummin*| **at** *hours:min*[[*month*] *day-of-month*] [*day-of-week*]} {**oneshot**| **recurring**| **system-startup**}
7. **policy-list** *list-name*
8. **exit**
9. **show kron schedule**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**kron policy-list** *list-name* [**conditional**] **Example:** `Router(config)# kron policy-list cns-weekly`	Specifies a name for a new or existing Command Scheduler policy list and enters kron-policy configuration mode. • If the *list-name* is new, a new policy list structure is created. • If the *list-name* exists, the existing policy list structure is accessed. The policy list is run in configured order with no editor function. • If the optional **conditional** keyword is used, execution of the commands stops when an error is encountered.
Step 4	**cli** *command* **Example:** `Router(config-kron-policy)# cli cns image retrieve server https://10.19.2.3/cnsweek/ status https://10.19.2.3/cnsstatus/week/`	Specifies the fully-qualified EXEC command and associated syntax to be added as an entry in the specified Command Scheduler policy list. • Each entry is added to the policy list in the order in which it is configured. • Repeat this step to add other EXEC CLI commands to a policy list to be executed at the same time or interval. **Note** EXEC commands that generate a prompt or can be terminated using keystrokes will cause an error.
Step 5	**exit** **Example:** `Router(config-kron-policy)# exit`	Exits kron-policy configuration mode and returns the router to global configuration mode.

	Command or Action	Purpose
Step 6	**kron occurrence** *occurrence-name* [**user** *username*] {**in**[[*numdays:*]*numhours:*]*nummin*| **at** *hours:min*[[*month*] *day-of-month*] [*day-of-week*]} {**oneshot**| **recurring**| **system-startup**} **Example:** `Router(config)# kron occurrence may user sales at 6:30 may 20 oneshot`	Specifies a name and schedule for a new or existing Command Scheduler occurrence and enters kron-occurrence configuration mode. • Use the **in** keyword to specify a delta time interval with a timer that starts when this command is configured. • Use the **at** keyword to specify a calendar date and time. • Choose either the **oneshot** or **recurring** keyword to schedule Command Scheduler occurrence once or repeatedly. Add the optional **system-startup** keyword for the occurrence to be at system startup.
Step 7	**policy-list** *list-name* **Example:** `Router(config-kron-occurrence)# policy-list sales-may`	Specifies a Command Scheduler policy list. • Each entry is added to the occurrence list in the order in which it is configured. **Note** If the CLI commands in a policy list generate a prompt or can be terminated using keystrokes, an error will be generated and the policy list will be deleted.
Step 8	**exit** **Example:** `Router(config-kron-occurrence)# exit`	Exits kron-occurrence configuration mode and returns the router to global configuration mode. • Repeat this step to exit global configuration mode.
Step 9	**show kron schedule** **Example:** `Router# show kron schedule`	(Optional) Displays the status and schedule information of Command Scheduler occurrences.

Examples

In the following example, output information is displayed about the status and schedule of all configured Command Scheduler occurrences:

```
Router# show kron schedule
Kron Occurrence Schedule
cns-weekly inactive, will run again in 7 days 01:02:33
may inactive, will run once in 32 days 20:43:31 at 6:30 on May 20
```

Troubleshooting Tips

Use the **debug kron** command in privileged EXEC mode to troubleshoot Command Scheduler command operations. Use any debugging command with caution because the volume of output generated can slow or stop the router operations.

Configuring Advanced Cisco Networking Services Features

Perform this task to configure more advanced Cisco Networking Services features. After the Cisco Networking Services agents are operational, you can configure some other features. You can enable the Cisco Networking Services inventory agent--that is, send an inventory of the router's line cards and modules to the Cisco Networking Services configuration engine--and enter Cisco Networking Services inventory mode.

Some other advanced features allow you to use the Software Developer's Toolkit (SDK) to specify how Cisco Networking Services notifications should be sent or how to access MIB information. Two encapsulation methods can be used: either nongranular (SNMP) encapsulation or granular (XML) encapsulation.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns mib-access encapsulation** {**snmp** | **xml**[**size** *bytes*]}
4. **cns notifications encapsulation** {**snmp** | **xml**}
5. **cns inventory**
6. **transport event**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns mib-access encapsulation** {**snmp**	**xml**[**size** *bytes*]} **Example:** `Router(config)# cns mib-access encapsulation snmp`	(Optional) Specifies the type of encapsulation to use when accessing MIB information. • Use the **snmp** keyword to specify that nongranular encapsulation is used to access MIB information. • Use the **xml** keyword to specify that granular encapsulation is used to access MIB information. The optional **size** keyword specifies the maximum size for response events, in bytes. The default byte value is 3072.

	Command or Action	Purpose
Step 4	**cns notifications encapsulation** {**snmp** | **xml**} **Example:** `Router(config)# cns notifications encapsulation xml`	(Optional) Specifies the type of encapsulation to use when sending Cisco Networking Services notifications. • Use the **snmp** keyword to specify that nongranular encapsulation is used when Cisco Networking Services notifications are sent. • Use the **xml** keyword to specify that granular encapsulation is used when Cisco Networking Services notifications are sent.
Step 5	**cns inventory** **Example:** `Router(config)# cns inventory`	Enables the Cisco Networking Services inventory agent and enters Cisco Networking Services inventory mode. • An inventory of the router's line cards and modules is sent to the Cisco Networking Services configuration engine.
Step 6	**transport event** **Example:** `Router(cns-inv)# transport event`	Specifies that inventory requests are sent out with each Cisco Networking Services inventory agent message.
Step 7	**exit** **Example:** `Router(cns-inv)# exit`	Exits Cisco Networking Services inventory mode and returns to global configuration mode. • Repeat this command to return to privileged EXEC mode.

Troubleshooting Cisco Networking Services Agents

This section explains how to troubleshoot Cisco Networking Services agent issues.

The **show** commands created for the Cisco Networking Services image agent display information that is reset to zero after a successful reload of the device. Depending on the configuration of the image distribution process, the new image may not reload immediately. When a reload is not immediate or has failed, use the Cisco Networking Services image agent **show** commands to determine whether the image agent has connected to the image distribution server over HTTP or whether the image agent is receiving events from an application over the Cisco Networking Services Event Bus.

SUMMARY STEPS

1. **enable**
2. **show cns image status**
3. **clear cns image status**
4. **show cns image connections**
5. **show cns image inventory**
6. **debug cns image** [**agent**| **all**| **connection**| **error**]
7. **show cns event connections**
8. **show cns event subject** [*name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**show cns image status** **Example:** `Router# show cns image status`	(Optional) Displays information about the Cisco Networking Services image agent status.
Step 3	**clear cns image status** **Example:** `Router# clear cns image status`	(Optional) Clears Cisco Networking Services image agent status statistics.
Step 4	**show cns image connections** **Example:** `Router# show cns image connections`	(Optional) Displays information about Cisco Networking Services image management server HTTP or HTTPS connections.
Step 5	**show cns image inventory** **Example:** `Router# show cns image inventory`	(Optional) Displays inventory information about the Cisco Networking Services image agent. • This command displays a dump of XML that would be sent out in response to an image agent inventory request message. The XML output can be used to determine the information requested by an application.

	Command or Action	Purpose
Step 6	**debug cns image** [**agent**| **all**| **connection**| **error**] **Example:** `Router# debug cns image all`	(Optional) Displays debugging messages for Cisco Networking Services image agent services.
Step 7	**show cns event connections** **Example:** `Router# show cns event connections`	(Optional) Displays the status of the Cisco Networking Services event agent connection--such as whether it is connecting to the gateway, connected, or active--and to display the gateway used by the event agent and its IP address and port number.
Step 8	**show cns event subject** [*name*] **Example:** `Router# show cns event subject subject1`	(Optional) Displays a list of subjects of the Cisco Networking Services event agent that are subscribed to by applications.

- Examples, page 44

Examples

Sample Output for the show cns image status Command

In the following example, status information about the Cisco Networking Services image agent is displayed using the **show cns image status**privileged EXEC command:

```
Router# show cns image status
Last upgrade started at 11:45:02.000 UTC Mon May 6 2003
Last upgrade ended at 11:56:04.000 UTC Mon May 6 2003 status SUCCESS
Last successful upgrade ended at 11:56:04.000 UTC Mon May 6 2003
Last failed upgrade ended at 06:32:15.000 UTC Wed Apr 16 2003
Number of failed upgrades: 2
Number of successful upgrades: 6
 messages received: 12
 receive errors: 5
Transmit Status
  TX Attempts:4
    Successes:3         Failures 2
```

Sample Output for the show cns image connections Command

In the following example, information about the status of the Cisco Networking Services image management HTTP connections is displayed using the **show cns image connections**privileged EXEC command:

```
show cns image connections

CNS Image Agent:  HTTP connections
```

```
Connection attempts 1
never connected:0   Abrupt disconnect:0
Last successful connection at 11:45:02.000 UTC Mon May 6 2003
```

Sample Output for the show cns image inventory Command

In the following example, information about the Cisco Networking Services image agent inventory is displayed using the **show cns image inventory**privileged EXEC command:

```
show cns image inventory

Inventory Report
imageInventoryReport deviceName imageID Router /imageID hostName Router /ho
IOS (tm) C2600 Software (C2600-I-M), Experimental Version 12.3(20030414:081500)]
Copyright (c) 1986-2003 by cisco Systems, Inc.
Compiled Mon 14-Apr-03 02:03 by engineer /versionString imageFile tftp://10.25.2.1.
```

Sample Output for the debug cns image Command

In the following example, debugging messages for all Cisco Networking Services image agent services are displayed using the **debug cns image**privileged EXEC command. The Cisco Networking Services image agent in this example is connecting to an image server over HTTP. After connecting, the image server asks for an inventory of the Cisco IOS device.

```
Router# debug cns image all

All cns image debug flags are on
Router# cns image retrieve

May  7 06:11:42.175: CNS Image Agent: set EXEC lock
May  7 06:11:42.175: CNS Image Agent: received message from EXEC
May  7 06:11:42.175: CNS Image Agent: set session lock 1
May  7 06:11:42.175: CNS Image Agent: attempting to send to destination(http://
10.1.36.8:8080/imgsrv/xgate):
?xml version="1.0" encoding="UTF-8"? cnsMessageversion="1.0" senderCredentials userName
dvlpr-7200-6 /userName /senderCredentials
messageID dvlpr-7200-6_2 /messageID sessionControl imageSessionStart version="1.0"
initiatorInfotrigger EXEC/trigger initiatorCredentials userName dvlpr-7200-6/userName
/initiatorCredentials /initiatorInfo /imageSessionStart /sessionControl /cnsMessage
May  7 06:11:42.175: CNS Image Agent: clear EXEC lock
May  7 06:11:42.175: CNS Image Agent: HTTP message sent url:http://10.1.36.8:8080/imgsrv/
xgate
May  7 06:11:42.191: CNS Image Agent: response data alloc 4096 bytes
May  7 06:11:42.191: CNS Image Agent: HTTP req data free
May  7 06:11:42.191: CNS Image Agent: response data freed
May  7 06:11:42.191: CNS Image Agent: receive message
?xml version="1.0" encoding="UTF-8"?
cnsMessage version="1.0"
senderCredentials
userName myImageServer.cisco.com/userName
passWord R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b/passWord
/senderCredentials
messageID dvlpr-c2600-2-476456/messageID
request
replyTo
serverReply http://10.1.36.8:8080/imgsrv/xgate /serverReply
/replyTo
imageInventory
inventoryItemList
all/
/inventoryItemList
/imageInventory
/request
/cnsMessage
```

Sample Output for the show cns event Commands

The following example displays the IP address and port number of the primary and backup gateways:

```
Router# show cns event connections

The currently configured primary event gateway:
        hostname is 10.1.1.1.
        port number is 11011.
Event-Id is Internal test1
Keepalive setting:
        none.
Connection status:
        Connection Established.
The currently configured backup event gateway:
        none.
The currently connected event gateway:
        hostname is 10.1.1.1.
        port number is 11011.
```

The following sample displays a list of subjects of the Cisco Networking Services event agent that are subscribed to by applications:

```
Router# show cns event subject

The list of subjects subscribed by applications.
   cisco.cns.mibaccess:request
   cisco.cns.config.load
   cisco.cns.config.reboot
   cisco.cns.exec.cmd
```

Configuration Examples for Cisco Networking Services

Deploying the Cisco Networking Services Router Example

The following example shows an initial configuration on a remote router. The hostname of the remote router is the unique ID. The Cisco Networking Services configuration engine IP address is 172.28.129.22.

```
cns template connect template1
 cli ip address negotiated
 cli encapsulation ppp
 cli ip directed-broadcast
 cli no keepalive
 cli no shutdown
 exit
cns connect host1 retry-interval 30 retries 3
exit
 hostname RemoteRouter
 ip route 172.28.129.22 255.255.255.0 10.11.11.1
 cns id Ethernet 0 ipaddress
```

```
cns config initial 10.1.1.1 no-persist
exit
```

Configuring a Partial Configuration Example

Incremental or partial configuration allows the remote router to be incrementally configured after its initial configuration. You must perform these configurations manually through the Cisco Networking Services configuration engine. The registrar allows you to change the configuration templates, edit parameters, and submit the new configuration to the router without a software or hardware restart.

The following example shows incremental (partial) configuration on a remote router. The Cisco Networking Services configuration engine IP address is 172.28.129.22, and the port number is 80.

```
cns config partial 172.28.129.22 80
```

Enabling and Configuring Cisco Networking Services Agents Example

The following example shows various Cisco Networking Services agents being enabled and configured starting with the configuration agent being enabled with the **cns config partial** command to configure an incremental (partial) configuration on a remote router. The Cisco Networking Services configuration engine IP address is 172.28.129.22, and the port number is 80. The Cisco Networking Services exec agent is enabled with an IP address of 172.28.129.23, and the Cisco Networking Services event agent is enabled with an IP address of 172.28.129.24. Until the Cisco Networking Services event agent is enabled, no other Cisco Networking Services agents are operational.

```
cns config partial 172.28.129.22 80
cns exec 172.28.129.23 source 172.22.2.2
cns event 172.28.129.24 source 172.22.2.1
exit
```

In the following example, the Cisco Networking Services image agent parameters are configured using the CLI. An image ID is specified to use the IP address of the FastEthernet interface 0/1, a password is configured for the Cisco Networking Services image agent services, the Cisco Networking Services image upgrade retry interval is set to four minutes, and image management and status servers are configured.

```
cns id FastEthernet0/1 ipaddress image
cns image retry 240
cns image password abctext
cns image server https://10.21.2.3/cns/imgsvr status https://10.21.2.3/cns/status/
```

In the following example, the Cisco Networking Services image agent is configured to use the Cisco Networking Services Event Bus. An image ID is specified as the hardware serial number of the networking device, the Cisco Networking Services event agent is enabled with a number of parameters, and the Cisco Networking Services image agent is enabled without any keywords or options. The Cisco Networking Services image agent will listen for events on the Cisco Networking Services Event Bus.

```
cns id hardware-serial image
cns event 10.21.9.7 11011 keepalive 240 120 failover-time 5
cns image
cns image password abctext
```

Cisco Networking Services Flow-Through Provisioning Examples

Cisco Configuration Express File Using T1 over HDLC Protocol Example

The following example shows use of the Cisco Configuration Express file to configure the remote router before delivery to its final premises. In the example, 172.28.129.22 is the IP address of the Cisco Networking Services configuration engine.

```
cns config initial 172.28.129.22 no-persist
!cns configure and event agents
cns event 172.28.129.22
controller t1 0
!T1 configuration
framing esf
linecode b8zs
channel-group 0 timeslots 1-24 speed 64
exit
cns id s0:0 ipaddress
interface s0:0
!Assigns IP address to s0:0
ip address slarp retry 2
exit
ip route 10.0.0.0 0.0.0.0 s0:0
!IP static route
end
```

T1 Configuration Template Example

The following example shows use of the T1 configuration template to build the configuration for use on T1:

```
hostname ${LDAP://this:attrName=IOShostname}
enable password ${LDAP://this:attrName=IOSpassword}
controller T1 0
clock source ${LDAP://this:attrName=IOST1-clocksource}
linecode ${LDAP://this:attrName=IOST1-line}
framing ${LDAP://this:attrName=IOST1-framing}
channel-group ${LDAP://this:attrName=IOST1-channel-group}
timeslots ${LDAP://this:attrName=IOST1-timeslots}
speed ${LDAP://this:attrName=IOST1-speed}
```

Voice Configuration Template Example

The following example shows use of the voice configuration template to build the configuration for using voice:

```
voice-port 1/1
codec ${LDAP://this:attrName=IOSvoice-port1}
exit
dial-peer voice 1 pots
application ${LDAP://this:attrName=IOSdial-peer1}
port 1/1
```

Remote Router Example

The following example shows a remote router configuration:

```
Router# show running-config
Current configuration: 1659 bytes
!
version 12.2
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname tira-24V
!
```

```
!
network-clock base-rate 64k
ip subnet-zero
ip cef
!
ip audit notify log
ip audit po max-events 100
!
class-map match-any voice
match access-group 100
!
!
policy-map qos
class voice
priority percent 70
voice service voip
h323
!
no voice confirmation-tone
voice-card 0
!
!
controller T1 0
framing sf
linecode ami
!
controller T1 1
mode cas
framing esf
linecode b8zs
ds0-group 0 timeslots 1 type e&m-immediate-start
ds0-group 1 timeslots 2 type e&m-immediate-start
!
!
interface Ethernet0
ip address 10.1.1.2 255.255.0.0
!
interface Serial0
bandwidth 1536
ip address 10.11.11.1 255.255.255.0
no ip mroute-cache
load-interval 30
clockrate 148000
!
ip classless
ip route 223.255.254.254 255.255.255.0 10.3.0.1
!
no ip http server
ip pim bidir-enable
!
access-list 100 permit udp any range 16384 32767 any
access-list 100 permit tcp any any eq 1720
call rsvp-sync
!
voice-port 1:0
timeouts wait-release 3
!
voice-port 1:1
timeouts wait-release 3
!
!
mgcp profile default
!
dial-peer cor custom
!
dial-peer voice 1000 pots
destination-pattern 1000
port 1:0
forward-digits 0
!
dial-peer voice 1001 pots
destination-pattern 1001
no digit-strip
```

```
port 1:1
forward-digits 0
!
dial-peer voice 2000 voip
destination-pattern 2000
session target ipv4:10.11.11.2
codec g711ulaw
!
dial-peer voice 2001 voip
destination-pattern 2001
session target ipv4:10.11.11.2
signal-type ext-signal
codec g711ulaw
!
!
line con 0
line aux 0
line 2 3
line vty 0 4
```

The following example shows configuration of a serial interface to connect to and download a configuration from a Cisco IE2100 Cisco Networking Services configuration engine. The IE2100 IP address is 10.1.1.1. The gateway IP address to reach the 10.1.1.0 network is 10.11.11.1. The Cisco Networking Services default ID is the hostname, so that **cns id** command is not needed. However, the **hostname** command is key to retrieving the configuration file on the Cisco Networking Services configuration engine.

This configuration auto-tries ever serial interface on the remote router in turn, applies the **config-cli** commands to that interface, and tries to ping the address in the **cns config initial** command. When it succeeds, it performs a normal initial configuration.

```
! Initial basic configuration (serial interface) PPP
cns connect serial retry-interval 1 retries 1
config-cli ip address negotiated
config-cli encapsulation ppp
config-cli ip directed-broadcast
config-cli no keepalive
config-cli no shutdown
exit
hostname 26ML
ip route 10.1.1.1 255.255.255.0 10.11.11.1
cns config initial 10.1.1.1 no-persist
cns inventory config
! Initial basic configuration (serial interface) HDLC
cns config connect serial retry-interval 1 retries 1
config-cli ip address slarp retry 1
config-cli no shutdown
exit
hostname tira-36V
ip route 10.1.1.1 255.255.255.0 10.11.11.1
cns config initial 10.1.1.1 no-persist
cns inventory config
Incremental configuration (serial interface)
cns config partial 10.1.1.1
cns event 10.1.1.1
```

Command Scheduler Policy Lists and Occurrences Examples

In the following example, a Command Scheduler policy named cns-weekly is configured to run two sets of EXEC CLI involving Cisco Networking Services commands. The policy is then scheduled with two other policies to run every seven days, one hour and thirty minutes.

```
kron policy-list cns-weekly
cli cns image retrieve server http://10.19.2.3/week/ status http://10.19.2.5/status/week/
cli cns config retrieve page /testconfig/config.asp no-persist
exit
kron occurrence week in 7:1:30 recurring
```

```
policy-list cns-weekly
policy-list itd-weekly
policy-list mkt-weekly
```

In the following example, a Command Scheduler policy named sales-may is configured to run a Cisco Networking Services command to retrieve a specified image from a remote server. The policy is then scheduled to run only once on May 20, at 6:30 a.m.

```
kron policy-list sales-may
cli cns image retrieve server 10.19.2.3 status 10.19.2.3
exit
kron occurrence may at 6:30 May 20 oneshot
policy-list sales-may
```

In the following example, a Command Scheduler policy named image-sunday is configured to run a Cisco Networking Services command to retrieve a specified image from a remote server. The policy is then scheduled to run every Sunday at 7:30 a.m.

```
kron policy-list image-sunday
cli cns image retrieve server 10.19.2.3 status 10.19.2.3
exit
kron occurrence sunday user sales at 7:30 sunday recurring
policy-list image-sunday
```

In the following example, a Command Scheduler policy named file-retrieval is configured to run a Cisco Networking Services command to retrieve a specific file from a remote server. The policy is then scheduled to run on system startup.

```
kron policy-list file-retrieval
cli cns image retrieve server 10.19.2.3 status 10.19.2.3
exit
kron occurrence system-startup
policy-list file-retrieval
```

Retrieving a Cisco Networking Services Image from a Server Example

In the following example, the Cisco Networking Services image agent polls a file server using the **cns image retrieve** command. Assuming that the Cisco Networking Services image agent is already enabled, the file server and status server paths specified here will overwrite any existing image agent server and status configuration. The new file server will be polled and a new image, if it exists, will be downloaded to the networking device.

```
cns image retrieve server https://10.19.2.3/cns/ status https://10.19.2.3/cnsstatus/
```

Retrieving a Cisco Networking Services Configuration from a Server Examples

Retrieving Configuration Data from the Cisco Networking Services Trusted Server

The following example shows how to request a configuration from a trusted server at 10.1.1.1:

```
cns trusted-server all 10.1.1.1
exit
cns config retrieve 10.1.1.1
```

The following example shows how to request a configuration from a trusted server at 10.1.1.1 and to configure a Cisco Networking Services configuration retrieve interval using the **cns config retrieve** command:

```
cns trusted-server all 10.1.1.1
exit
cns config retrieve 10.1.1.1 retry 50 interval 1500
CNS Config Retrieve Attempt 1 out of 50 is in progress
Next cns config retrieve retry is in 1499 seconds (Ctrl-Shft-6 to abort this command).
..
00:26:40: %CNS-3-TRANSPORT: CNS_HTTP_CONNECTION_FAILED:10.1.1.1 -Process= "CNS config
retv", ipl= 0, pid= 43
```

00:26:40: %CNS-3-TRANSPORT: CNS_HTTP_CONNECTION_FAILED -Process= "CNS config retv", ipl= 0, pid= 43.....

```
cns config retrieve 10.1.1.1
```

Applying the Retrieved Data to the Running Configuration File

The following example shows how to check and apply configuration data retrieved from the server to running configuration file only. The Cisco Networking Services Configuration Agent will attempt to retrieve configuration data at 30-second intervals until the attempt is successful, or is unsuccessful five times in these attempts.

```
cns config retrieve 10.1.1.1 syntax-check no-persist retry 5 interval 30
```

Overwriting the Startup Configuration File with the Retrieved Data

The following example shows how to overwrite the startup configuration file with the configuration data retrieved from the server. The configuration data will not be applied to the running configuration.

```
cns config retrieve 10.1.1.1 syntax-check no-persist retry 5 interval 30
cns config retrieve 10.1.1.1 overwrite-startup
```

Using the Cisco Networking Services Zero Touch Solution Examples

Configuring PPP on a Serial Interface

The following example shows the bootstrap configuration for configuring PPP on a serial interface:

```
cns template connect ppp-serial
cli ip address negotiated
cli encapsulation ppp
cli ip directed-broadcast
cli no keepalive
exit
cns template connect ip-route
cli ip route 10.0.0.0 0.0.0.0 ${next-hop}
exit
cns connect serial-ppp ping-interval 1 retries 1
discover interface serial
template ppp-serial
template ip-route
exit
hostname 26ML
cns config initial 10.1.1.1 no-persist inventory
```

Configuring PPP on an Asynchronous Interface

The following example shows the bootstrap configuration for configuring PPP on an asynchronous interface:

```
cns template connect async
cli modem InOut
 .
 .
 .
exit
cns template connect async-interface
cli encapsulation ppp
cli ip unnumbered FastEthernet0/0
cli dialer rotary-group 0
exit
cns template connect ip-route
cli ip route 10.0.0.0 0.0.0.0 ${next-hop}
exit
cns connect async
discover line Async
template async
discover interface
template async-interface
template ip-route
exit
hostname async-example
cns config initial 10.1.1.1 no-persist inventory
```

Configuring HDLC on a Serial Interface

The following example shows the bootstrap configuration for configuring High-Level Data Link Control (HDLC) on a serial interface:

```
cns template connect hdlc-serial
cli ip address slarp retry 1
exit
cns template connect ip-route
cli ip route 0.0.0.0 0.0.0.0 ${next-hop}
exit
cns connect hdlc-serial ping-interval 1 retries 1
discover interface serial
template hdlc-serial
template ip-route
exit
hostname host1
cns config initial 10.1.1.1 no-persist inventory
```

Configuring Aggregator Router Interfaces

The following examples show how to configure a standard serial interface and a serial interface bound to a controller on an aggregator router (also known as the DCE). In order for connectivity to be established, the aggregator router must have a point-to-point subinterface configured.

Standard Serial Interface

```
interface Serial0/1
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
exit
interface Serial0/1.1 point-to-point
 10.0.0.0 255.255.255.0
 frame-relay interface-dlci 8
```

Serial Interface Bound to a Controller

```
controller T1 0
 framing sf
 linecode ami
 channel-group 0 timeslots 1-24
exit
interface Serial0:0
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
exit
interface Serial0:0.1 point-to-point
 ip address ip-address mask
 frame-relay interface-dlci dlci
```

Configuring IP over Frame Relay

The following example shows the bootstrap configuration for configuring IP over Frame Relay on a CPE router:

```
cns template connect setup-frame
 cli encapsulation frame-relay
 exit
cns template connect ip-over-frame
 cli frame-relay interface-dlci ${dlci}
 cli ip address dynamic
 exit
cns template connect ip-route
 cli ip route 10.0.0.0 0.0.0.0 ${next-hop}
 exit
cns connect ip-over-frame
 discover interface Serial
 template setup-frame
 discover dlci
 template ip-over-frame
 template ip-route
exit
cns config initial 10.1.1.1
```

Configuring IP over Frame Relay over T1

The following example shows the bootstrap configuration for configuring IP over Frame Relay over T1 on a CPE router:

```
cns template connect setup-frame
 cli encapsulation frame-relay
 exit
cns template connect ip-over-frame
 cli frame-relay interface-dlci ${dlci}
 cli ip address dynamic
 exit
cns template connect ip-route
 cli ip route 0.0.0.0 0.0.0.0 ${next-hop}
 exit
cns template connect t1-controller
 cli framing esf
 cli linecode b8zs
 cli channel-group 0 timeslots 1-24 speed 56
 exit
cns connect ip-over-frame-over-t1
 discover controller T1
 template t1-controller
 discover interface
 template setup-frame
 discover dlci
 template ip-over-frame
 template ip-route
```

```
exit
cns config initial 10.1.1.1
```

Additional References

The following sections provide references related to the Cisco Networking Services feature.

Related Documents

Related Topic	Document Title
Cisco Networking Services commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Cisco Networking Services Configuration Engine	*Cisco Intelligence Engine 2100 Configuration Registrar Manual , Release 1.1 or later* Cisco Cisco Networking Services Configuration Engine Administrator's Guide

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
The CNS Flow-Through Provisioning feature provides two mechanisms for accessing MIBs: a nongranular mechanism using SNMP encapsulation and a granular mechanism using XML encapsulation. These mechanisms enable you to access the MIBS currently available in the remote router. The MIBS currently available depend on the router platform and Cisco IOS release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Cisco Networking Services

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 7 Feature Information for Cisco Networking Services

Feature Name	Releases	Feature Information
Cisco Networking Services	12.2(25)S 12.2(33) SRA 12.2(33)SB 12.2(33)SXI	The Cisco Networking Services feature is a collection of services that can provide remote event-driven configuring of Cisco IOS networking devices and remote execution of some command-line interface (CLI) commands. The following commands were introduced or modified by this feature: **clear cns config stats**, **clear cns counters**, **clear cns event stats**, **cli (cns)**, **cns config cancel**, **cns config initial**, **cns config notify**, **cns config partial**, **cns config retrieve**, **cns connect**, **cns event**, **cns exec**, **cns id**, **cns template connect**, **cns trusted-server**, **debug cns config**, **debug cns exec**, **debug cns xml-parser**, **logging cns-events**, **show cns config stats**, **show cns event connections**, **show cns event stats**, **show cns event subject**.

Feature Name	Releases	Feature Information
Cisco Networking Services Config Retrieve Enhancement with Retry and Interval	12.4(15)T 12.2(33)SRC 12.2(33)SB	The Cisco Networking Services Config Retrieve Enhancement with Retry and Interval feature adds two options to the **cns config retrieve** command enabling you to specify an amount of time in seconds to wait before attempting to retrieve a configuration from a trusted server. The number of retries is restricted to 100 to prevent the configuration agent from indefinitely attempting to reach an unreachable server. Use the keyboard combination **Ctrl-Shift-6** to abort the **cns config retrieve**command. • CNS Config Retrieve Enhancement with Retry and Interval, page 4 • Retrieving a CNS Configuration from a Server, page 27 • Retrieving a CNS Configuration from a Server: Example, page 43 The following command was modified by this feature: **cns config retrieve**.
Cisco Networking Services Enhanced Results Message	12.2(33)SRA 12.4(4)T	The Cisco Networking Services Enhanced Results Message feature sends a second Cisco Networking Services result message to the subject “cisco.cns.config.results” in addition to the Cisco Networking Services results messages sent to the Cisco Networking Services Event bus after a partial configuration is complete. The following command was modified by this feature: **cns config partial**.

Feature Name	Releases	Feature Information
Cisco Networking Services Event Agent	12.0(18)ST 12.0(22)S 12.2(2)T 12.2(33)SRA 12.2(33)SB 12.2(33)SXI	The Cisco Networking Services Event Agent is part of the Cisco IOS infrastructure that allows Cisco IOS applications to publish and subscribe to events on a Cisco Networking Services Event Bus. Cisco Networking Services Event Agent works in conjunction with the Cisco Networking Services Configuration Agent feature. The following commands were introduced or modified by this feature: **cns event**, **show cns event connections**, **show cns event stats**, **show cns event subject**.

Feature Name	Releases	Feature Information
Cisco Networking Services Flow-Through Provisioning	12.2(2)T 12.2(2)XB 12.2(11)YT 12.2(11)YV	Cisco Networking Services Flow-Through Provisioning provides the infrastructure for automated configuration of large numbers of network devices. Based on Cisco Networking Services event and config agents, it eliminates the need for an onsite technician to initialize the device. The result is an automated workflow from initial subscriber-order entry through Cisco manufacturing and shipping to final device provisioning and subscriber billing. This focuses on a root problem of today's service-provider and other similar business models: use of human labor in activating service. The following commands were introduced or modified by this feature: **cns config cancel**, **cns config connect-intf**, **cns config initial**, **cns config partial**, **cns config notify**, **cns event**, **cns id**, **cns inventory**, **cns mib-access encapsulation**, **cns notifications encapsulation**, **config-cli**, **debug cns config**, **debug cns event**, **debug cns management**, **debug cns xml-parser**, **line-cli**, **show cns config connections**, **show cns config outstanding**, **show cns event stats**, **show cns event subject**. **Note** The **cns config connect-intf**command was replaced by the **cns connect**and **cns template connect** commands in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.

Feature Name	Releases	Feature Information
		Note The **config-cli** and **line-cli** commands were replaced by the **cli (cns)** command in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.
Cisco Networking Services Frame-Relay Zero Touch	12.3(2)XF 12.3(8)T	The Cisco Networking Services Frame Relay Zero Touch feature provides a Cisco Networking Services zero touch deployment solution over Frame Relay where the CPE router discovers its DLCI and IP address dynamically and then contacts a Cisco Networking Services engine to retrieve its full configuration automatically. The following commands were introduced or modified by this feature: **cli (cns)**, **cns config connect-intf**, **cns connect, cns template connect**, **config-cli**, **discover (cns)**, **line-cli**, **template (cns)**. **Note** The **cns config connect-intf**command was replaced by the **cns connect**and **cns template connect** commands in Cisco IOS Releases 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases. **Note** The **config-cli** and **line-cli** commands were replaced by the **cli (cns)** command in Cisco IOS Releases 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.

Feature Name	Releases	Feature Information
Cisco Networking Services Image Agent	12.2(33)SEE 12.3(1) 12.2(31)SB2 12.2(33)SRB 12.2(33)SB 12.2(33)SXI	The Cisco Networking Services Image Agent feature is an infrastructure in Cisco IOS software to enable automated installation and activation of Cisco IOS images on Cisco IOS networking devices. The following commands were introduced or modified by this feature: **clear cns image connections**, **clear cns image status**, **cns id**, **cns image**, **cns image password**, **cns image retrieve**, **cns image retry**, **debug cns image**, **show cns image connections**, **show cns image inventory**, **show cns image status**.
Cisco Networking Services Interactive CLI	12.0(28)S 12.2(18)SXE 12.2(18)SXF2 12.2(33)SRC 12.2(33)SXI	The Cisco Networking Services Interactive CLI feature introduces a new XML interface that allows you to send interactive commands to a router, such as commands that generate prompts for user input.
Cisco Networking Services Security Enhancement	12.4(9)T 12.2(33)SRA	The Cisco Networking Services Security Enhancement feature improves the security of Cisco Networking Services messages by authenticating sender credentials through the use of the Service-Oriented Access Protocol (SOAP) message format. The following commands were introduced or modified by this feature: **cns aaa authentication**, **cns message format notification**.

Feature Name	Releases	Feature Information
Cisco Networking Services Zero Touch	12.3(9)	The Cisco Networking Services Zero Touch feature provides a zero touch deployment solution where the router contacts a Cisco Networking Services configuration engine to retrieve its full configuration automatically. The following commands were introduced or modified by this feature: **cli (cns)**, **cns config connect-intf**, **cns connect**, **cns template connect**, **config-cli**, **discover (cns)**, **line-cli, template (cns)**. **Note** The **cns config connect-intf**command was replaced by the **cns connect**and **cns template connect** commands in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases. **Note** The **config-cli** and **line-cli** commands were replaced by the **cli (cns)** command in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.
Command Scheduler	12.3(1) 12.2(33)SRA 12.2(33)SRC 12.2(33)SB 12.2(33)SXI	The Command Scheduler feature provides the ability to schedule some EXEC command-line interface (CLI) commands to run at specific times or at specified intervals. The following commands were introduced or modified by this feature: **cli**, **debug kron**, **kron occurrence**, **kron policy-list**, **policy-list**, **show kron schedule**.

Feature Name	Releases	Feature Information
DHCP Zero Touch	15.1(1)T	DHCP Option 43 allows you to configure the attributes of a device at initial deployment from a DHCP server. DCHP option 43 allows totally hands-free zero touch deployments for WSMA based deployments. The following commands were introduced: **wsma dhcp**, **cns dhcp**
CNS Configuration Agent	12.0(18)ST 12.0(22)S 12.2(2)T 12.2(8)T 12.2(33)SRA 12.2(33)SB 12.2(33)SXI	The Cisco Networking Services Configuration Agent feature supports routing devices by providing the following: • Initial configurations • Incremental (partial) configurations • Synchronized configuration updates The following commands were introduced or modified by this feature: **cns config cancel**, **cns config initial**, **cns config partial**, **cns config retrieve**, **cns password**, **debug cns config**, **debug cns xml-parser**, **show cns config outstanding, show cns config stats**, **show cns config status**.

Network Configuration Protocol

The Network Configuration Protocol (NETCONF) defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated. NETCONF uses Extensible Markup Language (XML)-based data encoding for the configuration data and protocol messages.

You can use the NETCONF over SSHv2 feature to perform network configurations via the Cisco command-line interface (CLI) over an encrypted transport. The NETCONF Network Manager, which is the NETCONF client, must use Secure Shell Version 2 (SSHv2) as the network transport to the NETCONF server. Multiple NETCONF clients can connect to the NETCONF server.

You can use the NETCONF over BEEP feature to send notifications of any configuration change over NETCONF. A notification is an event indicating that a configuration change has happened. The change can be a new configuration, deleted configuration, or changed configuration. The notifications are sent at the end of a successful configuration operation as one message showing the set of changes, rather than individual messages for each line in the configuration that is changed.

Blocks Extensible Exchange Protocol (BEEP) can use the Simple Authentication and Security Layer (SASL) profile to provide simple and direct mapping to the existing security model. Alternatively, NETCONF over BEEP can use the transport layer security (TLS) to provide a strong encryption mechanism with either server authentication or server and client-side authentication.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for NETCONF

- NETCONF over SSHv2 requires that a vty line be available for each NETCONF session as specified in the **netconf max-session**command.
- A vty line must be available for each NETCONF session as specified by the **netconf max-session** command.
- NETCONF over BEEP listeners require SASL to be configured.

Restrictions for NETCONF

- NETCONF SSHv2 supports a maximum of 16 concurrent sessions.
- Only SSH version 2 is supported.
- You must be running a crypto image in order to configure BEEP using TLS.

Information About NETCONF

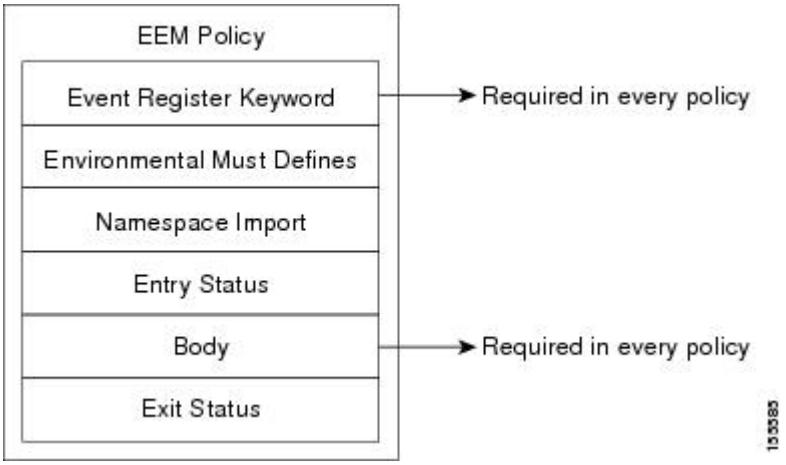
To configure NETCONF, you should understand the following concepts:

- NETCONF over SSHv2, page 66
- NETCONF over BEEP, page 67
- NETCONF Notifications, page 68

NETCONF over SSHv2

To run the NETCONF over SSHv2 feature, the client (a Cisco device running Cisco IOS software) establishes an SSH transport connection with the server (a NETCONF network manager). The following image shows a basic NETCONF over SSHv2 network configuration. The client and server exchange keys for security and password encryption. The user ID and password of the SSHv2 session running NETCONF are used for authorization and authentication purposes. The user privilege level is enforced and the client session may not have full access to the NETCONF operations if the privilege level is not high enough. If authentication, authorization, and accounting (AAA) is configured, the AAA service is used as if a user had established an SSH session directly to the device. Using the existing security configuration makes the transition to NETCONF almost seamless. Once the client has been successfully authenticated, the client invokes the SSH connection protocol and the SSH session is established. After the SSH session is established, the user or application invokes NETCONF as an SSH subsystem called “netconf.”

Figure 3



Secure Shell Version 2

SSHv2 runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. SSHv2 provides a means to securely access and securely execute commands on another computer over a network.

NETCONF does not support SSH version 1. The configuration for the SSH Version 2 server is similar to the configuration for SSH version 1. Use the **ip ssh version** command to specify which version of SSH that you want to configure. If you do not configure this command, SSH by default runs in compatibility mode; that is, both SSH version 1 and SSH version 2 connections are honored.

Note SSH version 1 is a protocol that has never been defined in a standard. If you do not want your router to fall back to the undefined protocol (version 1), you should use the **ip ssh version** command and specify version 2.

Use the **ip ssh rsa keypair-name** command to enable an SSH connection using Rivest, Shamir, and Adelman (RSA) keys that you have configured. If you configure the **ip ssh rsa keypair-name** command with a key-pair name, SSH is enabled if the key pair exists, or SSH will be enabled if the key pair is generated later. If you use this command to enable SSH, you do not need to configure a hostname and a domain name.

NETCONF over BEEP

The NETCONF over BEEP feature allows you to enable BEEP as the transport protocol to use during NETCONF sessions. Using NETCONF over BEEP, you can configure either the NETCONF server or the NETCONF client to initiate a connection, thus supporting large networks of intermittently connected devices, and those devices that must reverse the management connection where there are firewalls and Network Address Translators (NATs).

BEEP is a generic application protocol framework for connection-oriented, asynchronous interactions. It is intended to provide the features that traditionally have been duplicated in various protocol implementations. BEEP typically runs on top of TCP and allows the exchange of messages. Unlike HTTP and similar protocols, either end of the connection can send a message at any time. BEEP also includes facilities for encryption and authentication and is highly extensible.

The BEEP protocol contains a framing mechanism that permits simultaneous and independent exchanges of messages between peers. These messages are usually structured using XML. All exchanges occur in the context of a binding to a well-defined aspect of the application, such as transport security, user authentication, or data exchange. This binding forms a channel; each channel has an associated profile that defines the syntax and semantics of the messages exchanged.

The BEEP session is mapped onto the NETCONF service. When a session is established, each BEEP peer advertises the profiles it supports. During the creation of a channel, the client (the BEEP initiator) supplies one or more proposed profiles for that channel. If the server (the BEEP listener) creates the channel, it selects one of the profiles and sends it in a reply. The server may also indicate that none of the profiles are acceptable, and decline creation of the channel.

BEEP allows multiple data exchange channels to be simultaneously in use.

Although BEEP is a peer-to-peer protocol, each peer is labelled according to the role it is performing at a given time. When a BEEP session is established, the peer that awaits new connections is the BEEP listener. The other peer, which establishes a connection to the listener, is the BEEP initiator. The BEEP peer that starts an exchange is the client, and the other BEEP peer is the server. Typically, a BEEP peer that acts in the server role also performs in the listening role. However, because BEEP is a peer-to-peer protocol, the BEEP peer that acts in the server role is not required to also perform in the listening role.

Simple Authentication and Security Layer

The SASL is an Internet standard method for adding authentication support to connection-based protocols. SASL can be used between a security appliance and an Lightweight Directory Access Protocol (LDAP) server to secure user authentication.

Transport Layer Security

The TLS is an application-level protocol that provides for secure communication between a client and server by allowing mutual authentication, the use of hash for integrity, and encryption for privacy. TLS relies upon certificates, public keys, and private keys.

Certificates are similar to digital ID cards. They prove the identity of the server to clients. Each certificate includes the name of the authority that issued it, the name of the entity to which the certificate was issued, the entity's public key, and time stamps that indicate the certificate's expiration date.

Public and private keys are the ciphers used to encrypt and decrypt information. Although the public key is shared, the private key is never given out. Each public-private key pair works together. Data encrypted with the public key can be decrypted only with the private key.

Access Lists

You can optionally configure access lists for use with NETCONF over SSHv2 sessions. An access list is a sequential collection of permit and deny conditions that apply to IP addresses. The Cisco IOS software tests addresses against the conditions in an access list one by one. The first match determines whether the software accepts or rejects the address. Because the software stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the software rejects the address.

The two main tasks involved in using access lists are as follows:

1. Creating an access list by specifying an access list number or name and access conditions.
2. Applying the access list to interfaces or terminal lines.

For more information about configuring access lists, see IP Access List Overview and Creating an IP Access List and Applying It to an Interface modules in the Cisco IOS Security Configuration Guide: Securing the Data Plane .

NETCONF Notifications

NETCONF sends notifications of any configuration change over NETCONF. A notification is an event indicating that a configuration change has occurred. The change can be a new configuration, deleted configuration, or changed configuration. The notifications are sent at the end of a successful configuration operation as one message that shows the set of changes rather than showing individual messages for each line that is changed in the configuration.

How to Configure NETCONF

Enabling SSH Version 2 Using a Hostname and Domain Name

Perform this task to configure your router for SSH version 2 using a hostname and domain name. You may also configure SSH version 2 by using the RSA key pair configuration (see Enabling SSH Version 2 Using RSA Key Pairs, page 70).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hostname** *hostname*
4. **ip domain-name** *name*
5. **crypto key generate rsa**
6. **ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*]
7. **ip ssh version 2**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**hostname** *hostname* **Example:** `Router(config)# hostname host1`	Configures a hostname for your router.

	Command or Action	Purpose
Step 4	**ip domain-name** *name* **Example:** `Router(config)# ip domain-name domain1.com`	Configures a domain name for your router.
Step 5	**crypto key generate rsa** **Example:** `Router(config)# crypto key generate rsa`	Enables the SSH server for local and remote authentication.
Step 6	**ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*] **Example:** `Router(config)# ip ssh timeout 120`	(Optional) Configures SSH control variables on your router.
Step 7	**ip ssh version 2** **Example:** `Router(config)# ip ssh version 2`	Specifies the version of SSH to be run on your router.

Enabling SSH Version 2 Using RSA Key Pairs

Perform this task to enable SSH version 2 without configuring a hostname or domain name. SSH version 2 will be enabled if the key pair that you configure already exists or if it is generated later. You may also configure SSH version 2 by using the hostname and domain name configuration. (See "Enabling SSH Version 2 Using a Hostname and Domain Name, page 69.)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh rsa keypair-name** *keypair-name*
4. **crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size*
5. **ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*]
6. **ip ssh version 2**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip ssh rsa keypair-name** *keypair-name* **Example:** `Router(config)# ip ssh rsa keypair-name sshkeys`	Specifies which RSA keypair to use for SSH usage. **Note** A Cisco IOS router can have many RSA key pairs.
Step 4	**crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size* **Example:** `Router(config)# crypto key generate rsa usage-keys label sshkeys modulus 768`	Enables the SSH server for local and remote authentication on the router. For SSH version 2, the modulus size must be at least 768 bits. **Note** To delete the RSA key pair, use the **crypto key zeroize rsa** command. After you have deleted the RSA command, you automatically disable the SSH server.
Step 5	**ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*] **Example:** `Router(config)# ip ssh timeout 120`	Configures SSH control variables on your router.
Step 6	**ip ssh version 2** **Example:** `Router(config)# ip ssh version 2`	Specifies the version of SSH to be run on a router.

Starting an Encrypted Session with a Remote Device

Perform this task to start an encrypted session with a remote networking device. (You do not have to enable your router. SSH can be run in disabled mode.)

From any UNIX or UNIX-like device, the following command is typically used to form an SSH session:

```
ssh -2 -s user@router.example.com netconf
```

SUMMARY STEPS

1. Do one of the following:
   - **ssh** [**-v** {**1** | **2**}] [**-c** {**3des**| **aes128-cbc** | **aes192-cbc**| **aes256-cbc**}] [**-m**{**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*] {*ip-addr* | *hostname*} [*command*]

DETAILED STEPS

Command or Action	Purpose

Step 1 Do one of the following:
• **ssh** [**-v** {**1**
	2}] [**-c** {**3des**| **aes128-cbc**	**aes192-cbc**| **aes256-cbc**}] [**-m**{**hmac-md5**	**hmac-md5-96**	**hmac-sha1**	**hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*] {*ip-addr*	*hostname*} [*command*]
						Example:
						`Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-96 -l user2 10.76.82.24`
						Example:
						`Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-96 user2@10.76.82.24`
							Starts an encrypted session with a remote networking device.
							The first example adheres to the SSH version 2 conventions. A more natural and common way to start a session is by linking the username with the hostname. For example, the second configuration example provides an end result that is identical to that of the first example.

- Troubleshooting Tips, page 72
- What to Do Next, page 72

Troubleshooting Tips

The **ip ssh version** command can be used for troubleshooting your SSH configuration. By changing versions, you can determine which SSH version has a problem.

What to Do Next

For more information about the **ssh** command, see the Cisco IOS Security Command Reference.

Verifying the Status of the Secure Shell Connection

Perform this task to display the status of the SSH connection on your router.

Note You can use the following **show** commands in user EXEC or privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **show ssh**
3. **show ip ssh**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	(Optional) Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ssh** **Example:** `Router# show ssh`	Displays the status of SSH server connections.
Step 3	**show ip ssh** **Example:** `Router# show ip ssh`	Displays the version and configuration data for SSH.

Examples

The following output from the **show ssh** command displays status about SSH version 2 connections.

```
Router# show ssh
Connection Version Mode Encryption  Hmac               State
Username
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
%No SSHv1 server connections running.
```

The following output from the **show ip ssh** command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries.

```
Router# show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
```

Enabling NETCONF over SSHv2

Perform this task to enable NETCONF over SSHv2.

SSHv2 must be enabled.

Note There must be at least as many vty lines configured as there are concurrent NETCONF sessions.

Note

- A minimum of four concurrent NETCONF sessions must be configured.
- A maximum of 16 concurrent NETCONF sessions can be configured.
- NETCONF does not support SSHv1.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **netconf ssh** [**acl** *access-list-number*]
4. **netconf lock-time** *seconds*
5. **netconf max-sessions** *session*
6. **netconf max-message** size

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**netconf ssh** [**acl** *access-list-number*] **Example:** `Router(config)# netconf ssh acl 1`	Enables NETCONF over SSHv2. • Optionally, you can configure an access control list for this NETCONF session.
Step 4	**netconf lock-time** *seconds* **Example:** `Router(config)# netconf lock-time 60`	(Optional) Specifies the maximum time, in seconds, a NETCONF configuration lock is in place without an intermediate operation. • The valid range is 1 to 300. The default value is 10 seconds.

	Command or Action	Purpose
Step 5	**netconf max-sessions** *session* **Example:** `Router(config)# netconf max-sessions 5`	(Optional) Specifies the maximum number of concurrent NETCONF sessions allowed. • The valid range is 4 to 16. The default value is 4.
Step 6	**netconf max-message** size **Example:** `Router(config)# netconf max-message 37283`	(Optional) Specifies the maximum size, in kilobytes (KB), for the messages received in a NETCONF session. • The valid range is 1 to 2147483. The default value is infinite. • To set the maximum size to infinite, use the **no netconf max-message** command.

Configuring an SASL Profile

To enable NETCONF over BEEP using SASL, you must first configure an SASL profile, which specifies which users are allowed access into the router. Perform this task to configure an SASL profile.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sasl profile** *profile-name*
4. **mechanism di gest-md5**
5. **server** *user-name* **password** *password*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**sasl profile** *profile-name* **Example:** `Router(config)# sasl profile beep`	Configures an SASL profile and enters SASL profile configuration mode.
Step 4	**mechanism di gest-md5** **Example:** `Router(config-SASL-profile)# mechanism digest-md5`	Configures the SASL profile mechanism.
Step 5	**server** *user-name* **password** *password* **Example:** `Router(config-SASL-profile)# server user1 password password1`	Configures an SASL server.

Enabling NETCONF over BEEP

Perform this task to enable NETCONF over BEEP.

- There must be at least as many vty lines configured as there are concurrent NETCONF sessions.
- If you configure NETCONF over BEEP using SASL, you must first configure an SASL profile.

Note

- A minimum of four concurrent NETCONF sessions must be configured.
- A maximum of 16 concurrent NETCONF sessions can be configured.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa general-keys**
4. **crypto pki trustpoint** *name*
5. **enrollment url** *url*
6. **subject-name** *name*
7. **revocation-check** *method1* [*method2*[*method3*]]
8. **exit**
9. **crypto pki authenticate** *name*
10. **crypto pki enroll** *name*
11. **netconf lock-time** *seconds*
12. **line vty** line-number [*ending-line-number*]
13. **netconf max-sessions** *session*
14. **netconf beep initiator** {*hostname* | *ip-address*} *port-number* **user** *sasl-user* **password** *sasl-password*[**encrypt** *trustpoint*] [**reconnect-time** *seconds*]
15. **netconf beep listener** [*port-number*] [**acl** *access-list-number*] [**sasl** *sasl-profile*] [**encrypt** *trustpoint*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa general-keys** **Example:** `Router(config)# crypto key generate rsa general-keys`	Generates RSA key pairs and specifies that the general-purpose key pair should be generated. Perform this step only once.

	Command or Action	Purpose
Step 4	**crypto pki trustpoint** *name* **Example:** `Router(config)# crypto pki trustpoint my_trustpoint`	Declares the trustpoint that your router should use and enters ca-trustpoint configuration mode.
Step 5	**enrollment url** *url* **Example:** `Router(ca-trustpoint)# enrollment url http://10.2.3.3:80`	Specifies the enrollment parameters of a certification authority (CA).
Step 6	**subject-name** *name* **Example:** `Router(ca-trustpoint)# subject-name CN=dns_name_of_host.com`	Specifies the subject name in the certificate request. **Note** The subject name should be the Domain Name System (DNS) name of the device.
Step 7	**revocation-check** *method1* [*method2*[*method3*]] **Example:** `Router(ca-trustpoint)# revocation-check none`	Checks the revocation status of a certificate.
Step 8	**exit** **Example:** `Router(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.
Step 9	**crypto pki authenticate** *name* **Example:** `Router(config)# crypto pki authenticate my_trustpoint`	Authenticates the certification authority (by getting the certificate of the CA).
Step 10	**crypto pki enroll** *name* **Example:** `Router(config)# crypto pki enroll my_trustpoint`	Obtains the certificate or certificates for your router from CA.

Command or Action	Purpose
Step 11 **netconf lock-time** *seconds* **Example:** `Router(config)# netconf lock-time 60`	(Optional) Specifies the maximum time a NETCONF configuration lock is in place without an intermediate operation. The valid value range for the seconds argument is 1 to 300 seconds. The default value is 10 seconds.
Step 12 **line vty** line-number [*ending-line-number*] **Example:** `Router(config)# line vty 0 15`	Identifies a specific virtual terminal line for remote console access. You must configure the same number of vty lines as maximum NETCONF sessions.
Step 13 **netconf max-sessions** *session* **Example:** `Router(config)# netconf max-sessions 16`	(Optional) Specifies the maximum number of concurrent NETCONF sessions allowed.
Step 14 **netconf beep initiator** {*hostname* | *ip-address*} *port-number* **user** *sasl-user* **password** *sasl-password*[**encrypt** *trustpoint*] [**reconnect-time** *seconds*] **Example:** `Router(config)# netconf beep initiator host1 23 user user1 password password1 encrypt 23 reconnect-time 60`	(Optional) Specifies BEEP as the transport protocol for NETCONF sessions and configures a peer as the BEEP initiator. **Note** Perform this step to configure a NETCONF BEEP initiator session. You can also optionally configure a BEEP listener session.
Step 15 **netconf beep listener** [*port-number*] [**acl** *access-list-number*] [**sasl** *sasl-profile*] [**encrypt** *trustpoint*] **Example:** `Router(config)# netconf beep listener 26 acl 101 sasl profile1 encrypt 25`	(Optional) Specifies BEEP as the transport protocol for NETCONF and configures a peer as the BEEP listener. **Note** Perform this step to configure a NETCONF BEEP listener session. You can also optionally configure a BEEP initiator session.

Configuring the NETCONF Network Manager Application

SUMMARY STEPS

1. Use the following CLI string to configure the NETCONF Network Manager application to invoke NETCONF as an SSH subsystem:
2. As soon as the NETCONF session is established, indicate the server capabilities by sending an XML document containing a <hello>:
3. Use the following XML string to enable the NETCONF network manager application to send and receive NETCONF notifications:
4. Use the following XML string to stop the NETCONF network manager application from sending or receiving NETCONF notifications:

DETAILED STEPS

Step 1 Use the following CLI string to configure the NETCONF Network Manager application to invoke NETCONF as an SSH subsystem:

Example:

```
Unix Side: ssh-2 -s companyname@10.1.1.1 netconf
```

Step 2 As soon as the NETCONF session is established, indicate the server capabilities by sending an XML document containing a <hello>:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
    <hello>
      <capabilities>
        <capability>
            urn:ietf:params:xml:ns:netconf:base:1.0
          </capability>
          <capability>
            urn:ietf:params:ns:netconf:capability:startup:1.0
          </capability>
       </capabilities>
    <session-id>4<session-id>
</hello>]]>]]>
```

The client also responds by sending an XML document containing a <hello>:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello>
   <capabilities>
       <capability>
           urn:ietf:params:xml:ns:netconf:base:1.0
     </capability>
    </capabilities>
</hello>]]>]]>
```

Note Although the example shows the server sending a <hello> message followed by the client's message, both sides send the message as soon as the NETCONF subsystem is initialized, perhaps simultaneously.

Tip All NETCONF requests must end with]]>]]> which denotes an end to the request. Until the]]>]]> sequence is sent, the device will not process the request.

See "Configuring NETCONF over SSHv2 Example" for a specific example.

Step 3 Use the following XML string to enable the NETCONF network manager application to send and receive NETCONF notifications:

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.0"><notification-on/>
</rpc>]]>]]>
```

Step 4 Use the following XML string to stop the NETCONF network manager application from sending or receiving NETCONF notifications:

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.13"><notification-off/>
</rpc>]]>]]>
```

Delivering NETCONF Payloads

Use the following XML string to deliver the NETCONF payload to the network manager application:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.cisco.com/cpi_10/schema"
elementFormDefault="qualified" attributeFormDefault="unqualified" xmlns="http://
www.cisco.com/cpi_10/schema" xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!--The following elements define the cisco extensions for the content of the filter
element in a <get-config> request. They allow the client to specify the format of the
response and to select subsets of the entire configuration to be included.-->
   <xs:element name="config-format-text-block">
      <xs:annotation>
         <xs:documentation>If this element appears in the filter, then the cllient is
requesting that the response data be sent in config command block format.</
xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="text-filter-spec" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="config-format-text-cmd">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="text-filter-spec"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="config-format-xml">
      <xs:annotation>
         <xs:documentation>When this element appears in the filter of a get-config
```

```
request, the results are to be returned in E-DI XML format. The content of this element
is treated as a filter.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:complexContent>
            <xs:extension base="xs:anyType"/>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
   <!--These elements are used in the filter of a <get> to specify operational data to
return.-->
   <xs:element name="oper-data-format-text-block">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="show" type="xs:string" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="oper-data-format-xml">
      <xs:complexType>
         <xs:sequence>
            <xs:any/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <!--When confing-format-text format is specified, the following describes the content
of the data element in the response-->
   <xs:element name="cli-config-data">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="cmd" type="xs:string" maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>Content is a command. May be multiple lines.</
xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="cli-config-data-block" type="xs:string">
      <xs:annotation>
         <xs:documentation>The content of this element is the device configuration as it
would be sent to a terminal session. It contains embedded newline characters that must be
preserved as they represent the boundaries between the individual command lines</
xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="text-filter-spec">
      <xs:annotation>
         <xs:documentation>If this element is included in the config-format-text element,
then the content is treated as if the string was appended to the "show running-config"
command line.</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="cli-oper-data-block">
      <xs:complexType>
         <xs:annotation>
            <xs:documentation> This element is included in the response to get operation.
Content of this element is the operational data in text format.</xs:documentation>
         </xs:annotation>
         <xs:sequence>
            <xs:element name="item" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="show"/>
                     <xs:element name="response"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:schema>
```

Formatting NETCONF Notifications

The NETCONF network manager application uses .xsd schema files to describe the format of the XML NETCONF notification messages being sent between a NETCONF network manager application and a router running NETCONF over SSHv2 or BEEP. These files can be displayed in a browser or a schema reading tool. You can use these schema to validate that the XML is correct. These schema describe the format, not the content, of the data being exchanged.

NETCONF uses the <edit-config> function to load all of a specified configuration to a specified target configuration. When this new configuration is entered, the target configuration is not replaced. The target configuration is changed according to the data and requested operations of the requesting source.

The following are schemas for the NETCONF <edit-config> function in CLI, CLI block, and XML format.

NETCONF <edit-config> Request: CLI Format

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cli-config-data>
<cmd>hostname test</cmd>
            <cmd>interface fastEthernet0/1</cmd>
            <cmd>ip address 192.168.1.1 255.255.255.0</cmd>
</cli-config-data>
      </config>
   </edit-config>
</rpc>]]>]]>
```

NETCONF <edit-config> Response: CLI Format

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:netconf:base:1.0">
   <ok/>
</rpc-reply>]]>]]>
```

NETCONF <edit-config> Request: CLI-Block Format

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="netconf.mini.edit.3">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cli-config-data-block>
            hostname bob
            interface fastEthernet0/1
            ip address 192.168.1.1 255.255.255.0
         </cli-config-data-block>
      </config>
   </edit-config>
</rpc>]]>]]>
```

NETCONF <edit-config> Response: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="netconf.mini.edit.3" xmlns="urn:ietf:params:netconf:base:1.0">
```

```
   <ok/>
</rpc-reply>]]>]]>
```

The following are schemas for the NETCONF <get-config> function in CLI and CLI-block format.

NETCONF <get-config> Request: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <config-format-text-cmd>
            <text-filter-spec> | inc interface </text-filter-spec>
         </config-format-text-cmd>
</filter>
   </get-config>
</rpc>]]>]]>
```

NETCONF <get-config> Response: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <cli-config-data>
         <cmd>interface FastEthernet0/1</cmd>
         <cmd>interface FastEthernet0/2</cmd>
      </cli-config-data>
   </data>
</rpc-reply>]]>]]>
```

NETCONF <get-config> Request: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <config-format-text-block>
            <text-filter-spec> | inc interface </text-filter-spec>
         </config-format-text-block>
      </filter>
   </get-config>
</rpc>]]>]]>
```

NETCONF <get-config> Response: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <cli-config-data-block>
         interface FastEthernet0/1
         interface FastEthernet0/2
      </cli-config-data-block>
   </data>
</rpc-reply>]]>]]>
```

NETCONF uses the <get> function to retrieve configuration and device-state information. The NETCONF <get> format is the equivalent of a Cisco IOS **show** command. The <filter> parameter specifies the portion of the system configuration and device-state data to retrieve. If the <filter> parameter is empty, nothing is returned.

The following are schemas for the <get> function in CLI and CLI-block format.

NETCONF <get> Request: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
       <filter>
          <config-format-text-cmd>
             <text-filter-spec> | include interface </text-filter-spec>
          </config-format-text-cmd>
          <oper-data-format-text-block>
             <show>interfaces</show>
             <show>arp</show>
          </oper-data-format-text-block>
       </filter>
    </get>
 </rpc>]]>]]>
```

NETCONF <get> Response: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
       <cli-config-data>
<cmd>interface Loopback0</cmd>
<cmd>interface GigabitEthernet0/1</cmd>
<cmd>interface GigabitEthernet0/2</cmd>
</cli-config-data>
<cli-oper-data-block>
          <item>
             <show>interfaces</show>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          <show>arp</show>
          <item>
             <show>arp</show>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cli-oper-data-block>
    </data>
</rpc-reply>]]>]]>
```

NETCONF <get> Request: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
       <filter>
          <config-format-text-block>
             <text-filter-spec> | include interface </text-filter-spec>
          </config-format-text-block>
          <oper-data-format-text-block>
             <show>interfaces</show>
             <show>arp</show>
          </oper-data-format-text-block>
       </filter>
    </get>
 </rpc>]]>]]>
```

NETCONF <get> Response: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
```

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
       <cli-config-data-block>
interface Loopback0
interface GigabitEthernet0/1
interface GigabitEthernet0/2
       </cli-config-data-block>
       <cli-oper-data-block>
          <item>
             <show>interfaces</show>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          <show>arp</show>
          <item>
             <show>arp</show>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cli-oper-data-block>
    </data>
</rpc-reply>]]>]]>
```

Monitoring and Maintaining NETCONF Sessions

Perform this task to monitor and maintain NETCONF sessions.

Note

- A minimum of four concurrent NETCONF sessions must be configured.
- A maximum of 16 concurrent NETCONF sessions can be configured.
- NETCONF does not support SSHv1.

>

SUMMARY STEPS

1. **enable**
2. **show netconf** {**counters** | **session**| **schema**}
3. **debug netconf** {**all** | **error**}
4. **clear netconf** {**counters** | **sessions**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**show netconf** {**counters** | **session**| **schema**} **Example:** `Router# show netconf counters`	Displays NETCONF information.
Step 3	**debug netconf** {**all** | **error**} **Example:** `Router# debug netconf error`	Enables debugging of NETCONF sessions.
Step 4	**clear netconf** {**counters** | **sessions**} **Example:** `Router# clear netconf sessions`	Clears NETCONF statistics counters and NETCONF sessions, and frees associated resources and locks.

Configuration Examples for NETCONF

Enabling SSHv2 Using a Hostname and Domain Name Example

The following example shows how to configure SSHv2 using a hostname and a domain name:

```
Router# configure terminal
Router(config)# hostname host1
Router(config)# ip domain-name domain1.com
Router(config)# crypto key generate rsa
Router(config)# ip ssh timeout 120
Router(config)# ip ssh version 2
```

Enabling Secure Shell Version 2 Using RSA Keys Example

The following example shows how to configure SSHv2 using RSA keys:

```
Router# configure terminal

Router(config)# ip ssh rsa keypair-name sshkeys

Router(config)# crypto key generate rsa usage-keys label sshkeys modulus 768
Router(config)# ip ssh timeout 120
Router(config)# ip ssh version 2
```

Starting an Encrypted Session with a Remote Device Example

The following example shows how to start an encrypted SSH session with a remote networking device, from any UNIX or UNIX-like device:

```
Router(config)# ssh -2 -s user@router.example.com netconf
```

Configuring NETCONF over SSHv2 Example

The following example shows how to configure NETCONF over SSHv2:

```
Router# configure terminal
Router(config)# netconf ssh acl 1
Router(config)# netconf lock-time 60
Router(config)# netconf max-sessions 5
Router(config)# netconf max-message 2345
Router# ssh-2 -s username@10.1.1.1 netconf
```

The following example shows how to get the configuration for loopback interface 113.

SUMMARY STEPS

1. First, send the “hello”:
2. Next, send the get-config request:

DETAILED STEPS

Command or Action	Purpose

Step 1 First, send the “hello”:
 	
Example:
 	
`<?xml version="1.0" encoding=\"UTF-8\"?>`
`<hello><capabilities>`
`<capability>u?rn:ietf:params:netconf:base:1.0</capability>`
`<capability>urn:ietf:params:netconf:capability:writeable-running:1.0</capability>`
`<capability>urn:ietf:params:netconf:ca?pability:roll?back-on-error:1.0</capability>`
`<capability>urn:ietf:params:netconf:capability:startup:1.0</capability>`
`<capability>urn:ietf:params:netconf:ca?pability:url:?1.0</capability>`
`<capability>urn:cisco:params:netconf:capability:pi-data-model:1.0</capability>`
`<capability>urn:cisco:params:netconf:capabili?ty:notificati?on:1.0</capability>`
`</capabilities>`
`</hello>]]>]]>`	

Step 2 Next, send the get-config request:
 	
Example:
 	
`<?xml version="1.0"?>`
`<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"xmlns:cpi="http://www.cisco.com/cpi_10/`
`schema" message-id="101">`
`<get-config>`
`<source>`
`<running/>`
`</source>`
`<filter>`
`<config-format-text-cmd>`
`<text-filter-spec>`
`interface Loopback113`
`</text-filter-spec>`
`</config-format-text-cmd>`
`</filter>`
`</get-config>`
`</rpc>]]>]]>`	

The following output is shown on the router:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101"xmlns=\"urn:ietf:params:netconf:base:1.0\">
   <data>
      <cli-config-data>
interface Loopback113
description test456
no ip address
load-interval 30
end
       </cli-config-data>
    </data>
</rpc-rep?ly>]]>]]>
```

Configuring NETCONF over BEEP Example

The following example shows how to configure NETCONF over BEEP:

```
Router# configure terminal
Router(config)# crypto key generate rsa general-keys
```

```
Router(ca-trustpoint)# crypto pki trustpoint my_trustpoint

Router(ca-trustpoint)# enrollment url http://10.2.3.3:80
Router(ca-trustpoint)# subject-name CN=dns_name_of_host.com
Router(ca-trustpoint)# revocation-check none
Router(ca-trustpoint)# crypto pki authenticate my_trustpoint

Router(ca-trustpoint)# crypto pki enroll my_trustpoint

Router(ca-trustpoint)# line vty 0 15

Router(ca-trustpoint)# exit
Router(config)# netconf lock-time 60

Router(config)# netconf max-sessions 16

Router(config)# netconf beep initiator host1 23 user my_user password my_password encrypt
my_trustpoint reconnect-time 60

Router(config)# netconf beep listener 23 sasl user1 encrypt my_trustpoint
```

Configuring NETCONF Network Manager Application Example

The following example shows how to configure the NETCONF Network Manager application to invoke NETCONF as an SSH subsystem:

```
Unix Side: ssh-2 -s companyname@10.1.1.1 netconf
```

As soon as the NETCONF session is established, indicate the server capabilities by sending an XML document containing a <hello>:

```
<?xml version="1.0" encoding="UTF-8"?>
    <hello>
      <capabilities>
        <capability>
            urn:ietf:params:xml:ns:netconf:base:1.0
          </capability>
          <capability>
            urn:ietf:params:ns:netconf:capability:startup:1.0
          </capability>
       </capabilities>
    <session-id>4<session-id>
</hello>]]>]]>
```

The client also responds by sending an XML document containing a <hello>:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello>
   <capabilities>
       <capability>
           urn:ietf:params:xml:ns:netconf:base:1.0
     </capability>
    </capabilities>
</hello>]]>]]>
```

Use the following XML string to enable the NETCONF network manager application to send and receive NETCONF notifications:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.0"><notification-on/>
</rpc>]]>]]>
```

Use the following XML string to stop the NETCONF network manager application from sending or receiving NETCONF notifications:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<rpc message-id="9.13"><notification-off/>
</rpc>]]>]]>
```

Monitoring NETCONF Sessions Example

The following is sample output from the **show netconf counters** command:

```
Router# show netconf counters
NETCONF Counters
Connection Attempts:0: rejected:0 no-hello:0 success:0
Transactions
        total:0, success:0, errors:0
detailed errors:
        in-use 0        invalid-value 0         too-big 0
        missing-attribute 0     bad-attribute 0         unknown-attribute 0
        missing-element 0       bad-element 0   unknown-element 0
        unknown-namespace 0     access-denied 0         lock-denied 0
        resource-denied 0       rollback-failed 0       data-exists 0
        data-missing 0  operation-not-supported 0       operation-failed 0
        partial-operation 0
```

The following is sample output from the **show netconf session** command:

```
Router# show netconf session
(Current | max) sessions:   3 | 4
Operations received: 100              Operation errors: 99
Connection Requests: 5                Authentication errors: 2   Connection Failures: 0
ACL dropped : 30
Notifications  Sent: 20
```

The output of the **show netconf schema** command describes the element structure for a NETCONF request and the resulting reply. This schema can be used to construct proper NETCONF requests and parse the resulting replies. The nodes in the schema are defined in RFC 4741. The following is sample output from the **show netconf schema**command:

```
Router# show netconf schema
New Name Space 'urn:ietf:params:xml:ns:netconf:base:1.0'
<VirtualRootTag> [0, 1] required
  <rpc-reply> [0, 1] required
    <ok> [0, 1] required
    <data> [0, 1] required
    <rpc-error> [0, 1] required
      <error-type> [0, 1] required
      <error-tag> [0, 1] required
      <error-severity> [0, 1] required
      <error-app-tag> [0, 1] required
      <error-path> [0, 1] required
      <error-message> [0, 1] required
      <error-info> [0, 1] required
        <bad-attribute> [0, 1] required
        <bad-element> [0, 1] required
        <ok-element> [0, 1] required
        <err-element> [0, 1] required
        <noop-element> [0, 1] required
        <bad-namespace> [0, 1] required
        <session-id> [0, 1] required
  <hello> [0, 1] required
    <capabilities> 1 required
      <capability> 1+ required
  <rpc> [0, 1] required
    <close-session> [0, 1] required
    <commit> [0, 1] required
      <confirmed> [0, 1] required
      <confirm-timeout> [0, 1] required
    <copy-config> [0, 1] required
      <source> 1 required
        <config> [0, 1] required
          <cli-config-data> [0, 1] required
            <cmd> 1+ required
```

```
      <cli-config-data-block> [0, 1] required
      <xml-config-data> [0, 1] required
        <Device-Configuration> [0, 1] required
          <> any subtree is allowed
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
<delete-config> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
<discard-changes> [0, 1] required
<edit-config> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
  <default-operation> [0, 1] required
  <test-option> [0, 1] required
  <error-option> [0, 1] required
  <config> 1 required
    <cli-config-data> [0, 1] required
      <cmd> 1+ required
    <cli-config-data-block> [0, 1] required
    <xml-config-data> [0, 1] required
      <Device-Configuration> [0, 1] required
        <> any subtree is allowed
<get> [0, 1] required
  <filter> [0, 1] required
    <config-format-text-cmd> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-text-block> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-xml> [0, 1] required
    <oper-data-format-text-block> [0, 1] required
      <show> 1+ required
    <oper-data-format-xml> [0, 1] required
      <show> 1+ required
<get-config> [0, 1] required
  <source> 1 required
    <config> [0, 1] required
      <cli-config-data> [0, 1] required
        <cmd> 1+ required
      <cli-config-data-block> [0, 1] required
      <xml-config-data> [0, 1] required
        <Device-Configuration> [0, 1] required
          <> any subtree is allowed
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
  <filter> [0, 1] required
    <config-format-text-cmd> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-text-block> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-xml> [0, 1] required
<kill-session> [0, 1] required
  <session-id> [0, 1] required
<lock> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
```

```
    <url> [0, 1] required
<unlock> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
<validate> [0, 1] required
  <source> 1 required
    <config> [0, 1] required
      <cli-config-data> [0, 1] required
        <cmd> 1+ required
      <cli-config-data-block> [0, 1] required
      <xml-config-data> [0, 1] required
        <Device-Configuration> [0, 1] required
          <> any subtree is allowed
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
<notification-on> [0, 1] required
<notification-off> [0, 1] required
```

Additional References

The following sections provide references related to the NETCONF feature.

Related Documents

Related Topic	Document Title
IP access lists	IP Access List Overview and Creating an IP Access List and Applying It to an Interface modules in the Cisco IOS Security Configuration Guide: Securing the Data Plane.
Secure Shell and Secure Shell Version 2	“Configuring Secure Shell” module in the Cisco IOS Security Configuration Guide: Securing User Services.
NETCONF commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
IP access lists commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples Security commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2222	*Simple Authentication and Security Layer (SASL)*
RFC 2246	*The TLS Protocol Version 1.0*
RFC 3080	*The Blocks Extensible Exchange Protocol Core*
RFC 4251	*The Secure Shell (SSH) Protocol Architecture*
RFC 4252	*The Secure Shell (SSH) Authentication Protocol*
RFC 4741	NETCONF Configuration Protocol
RFC 4742	Using the NETCONF Configuration Protocol over Secure SHell (SSH)
RFC 4744	Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for NETCONF

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software

release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 8 Feature Information for NETCONF

Feature Name	Releases	Feature Information
NETCONF over SSHv2	12.2(33)SRA 12.4(9)T 12.2(33)SB 12.2(33)SXI	The NETCONF over SSHv2 feature enables you to perform network configurations via the Cisco command-line interface (CLI) over an encrypted transport. The NETCONF protocol defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated. NETCONF uses an Extensible Markup Language (XML)-based data encoding for the configuration data and protocol messages. • In 12.4(9)T, this feature was introduced. The following commands were introduced or modified by this feature: **clear netconf**, **debug netconf**, **netconf lock-time**, **netconf max-sessions**, **netconf ssh**, **show netconf**.

Feature Name	Releases	Feature Information
NETCONF Access for Configuration over BEEP	12.4(9)T 12.2(33)SRB 12.2(33)SB 12.2(33)SXI	The NETCONF over BEEP feature allows you to enable either the NETCONF server or the NETCONF client to initiate a connection, thus supporting large networks of intermittently connected devices and those devices that must reverse the management connection where there are firewalls and network address translators (NATs). • In 12.4(9)T, this feature was introduced. The following commands were introduced or modified by this feature: **netconf beep initiator**, **netconf beep listener**.

Glossary

BEEP --Blocks Extensible Exchange Protocol. A generic application protocol framework for connection-oriented, asynchronous interactions.

NETCONF --Network Configuration Protocol. A protocol that defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated.

SASL --Simple Authentication and Security Layer. An Internet standard method for adding authentication support to connection-based protocols. SASL can be used between a security appliance and an Lightweight Directory Access Protocol (LDAP) server to secure user authentication.

SSHv2 --Secure Shell Version 2. SSH runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. SSHv2 provides a means to securely access and securely execute commands on another computer over a network.

TLS --Transport Layer Security. An application-level protocol that provides for secure communication between a client and server by allowing mutual authentication, the use of hash for integrity, and encryption for privacy. TLS relies upon certificates, public keys, and private keys.

XML --Extensible Markup Language. A standard maintained by the World Wide Web Consortium (W3C) that defines a syntax that lets you create markup languages to specify information structures. Information structures define the type of information (for example, subscriber name or address), not how the information looks (bold, italic, and so on). External processes can manipulate these information structures and publish them in a variety of formats. XML allows you to define your own customized markup language.