Cisco Networking Services Configuration Guide, Cisco IOS XE Release 3E

First Published: August 26, 2013

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

CHAPTER 1

Network Configuration Protocol

The Network Configuration Protocol (NETCONF) defines a simple mechanism through which a network device can be managed, configuration data 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.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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

A vty line must be available for each NETCONF session as specified by the **netconf max-session** command.

Information About NETCONF

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

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 message from the client, 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 the "Example: Configuring NETCONF over SSHv2" section 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 client 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="exec"/>
                     <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 that are sent between a NETCONF network manager application and a device running NETCONF over SSHv2 or BEEP. These files can be displayed in a browser or a schema reading tool. You can use these schemas to validate that the XML is correct. These schemas 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>
             <exec>show interfaces</exec>
             <exec>show arp</exec>
          </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>
             <exec>show interfaces</exec>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          </item>
          <item>
             <exec>show arp</exec>
             <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>
             <exec>show interfaces</exec>
             <exec>show arp</exec>
          </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>
             <exec>show interfaces</exec>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          </item>
          <item>
             <exec>show arp</exec>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cli-oper-data-block>
    </data>
</rpc-reply>]]>]]>
```

Monitoring and Maintaining 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:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	**show netconf** {**counters** | **session**| **schema**}
	Example:
	`Device# show netconf counters`
		Displays NETCONF information.

Step 3	**debug netconf** {**all** | **error**}
	Example:
	`Device# debug netconf error`
		Enables debugging of NETCONF sessions.

Step 4	**clear netconf** {**counters** | **sessions**}
	Example:
	`Device# clear netconf sessions`
		Clears NETCONF statistics counters and NETCONF sessions, and frees associated resources and locks.

Configuration Examples for NETCONF

Example: Configuring the NETCONF Network Manager Application

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>]]>]]>
```

Example: Monitoring NETCONF Sessions

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

```
Device# 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:

```
Device# 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 displays 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:

```
Device# 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
      <exec> [0, 1] required
      <show> [0, 1] required
    <oper-data-format-xml> [0, 1] required
      <exec> [0, 1] required
      <show> [0, 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 for NETCONF

Related Documents

Related Topic	Document Title
Cicso IOS commands	Cisco IOS Master Command List, All Releases
NETCONF commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Cisco Networking Services Command Reference*
Security and IP access lists commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*

Standards and RFCs

Standard/RFC	Title
RFC 4251	*The Secure Shell (SSH) Protocol Architecture*
RFC 4252	*The Secure Shell (SSH) Authentication Protocol*
RFC 4741	*NETCONF Configuration Protocol*
RFC 4744	*Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

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 1: Feature Information for NETCONF

Feature Name	Releases	Feature Information
NETCONF XML PI	Cisco IOS XE 3.8S Cisco IOS XE 3.3SE	The NETCONF protocol was enhanced, adding format attribute support for all Cisco IOS exec commands. The following commands were modified: **clear netconf**, **debug netconf**, and **show netconf**.

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 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 a 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 appears (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.

CHAPTER 2

DHCP Zero Touch

The Cisco Dynamic Host Control Protocol (DHCP) Zero Touch feature enables a device to retrieve configuration files from the remote DHCP server during initial deployment with no end-user intervention.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Information About DHCP Zero Touch

DHCP Zero Touch Overview

The DHCP Zero Touch feature enables a device to retrieve configuration files from the remote Dynamic Host Configuration Protocol (DHCP) server during the initial device deployment without end-user intervention. You need a bootstrap configuration to communicate between the device 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 server. The DHCP Zero Touch feature introduces another method of retrieving bootstrap configuration information: using the DHCP Option 43 message. To accommodate situations where devices cannot have a pre-installed bootstrap configuration, a deployment model that uses DHCP Option 43 messages is used. Cisco recommends using DHCP Option 43 messages 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 preconfigured on the DHCP server. Within this DHCP Option 43 message, predefined parameterized commands are provided to the DHCP client. A timer for three minutes is set. After the timeout, if the file download is successful, the process is complete. If the file download fails, check the generated DHCP Option 43 message and correct any problems. Power cycle the device to retry the DHCP Option 43 message process.

Initiating DHCP Option 43 Messages with Cisco Networking Services

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

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

Cisco Networking Services Parameterized Commands

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 zero touch deployment (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

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 Services Command Reference.

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 zero :

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

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.
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 device tries 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	Parameter to CLI Mapping: 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`
				`Device(config)#` **`cns id string 881-ap event`**
				`Device(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`	`Device(config)#` **`cns config initial 10.10.10.1`**
CE config server port	C	(Optional) Specifies the numeric string values between 0 and 65535. The default value is 80.	`C11025`	`Device(config)#` **`cns config initial ce-address 11025`**
Source interface	D	(Optional) Indicates the source interface name.	`DF0/1`	`Device(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.	`E/cns/config.asp`	`Device(config)#` **`cns config initial ce-address page /cns/config.asp`**
Config no-persist	I		`I1`	`Device(config)#` **`cns config initial no-persist`**

Parameter	Letter Code	Values	Parameter to CLI Mapping: Sample Letter Code	Parameter to CLI Mapping: Sample CLI Mapping
		(Optional) Specifies the configuration conditions to NVRAM: 1-no-persist: Do not write configuration to NVRAM 1-persist: Write configuration to NVRAM Default—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 Passive Template (Cisco Networking Service Indicators)

Parameter	Letter Code	Values	Parameter to CLI Mapping: Sample Letter Code	Parameter to CLI Mapping: 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`
				`Device(config)#` **`cns id string 881-ap event`**
				`Device(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`	`Device(config)#` **`cns config initial 10.10.10.1`**
CE config server port	C	(Optional) Specifies the numeric string values between 0 and 65535. The default value is 80.	`C11025`	`Device(config)#` **`cns config initial ce-address 11025`**
Source interface	D	(Optional) Indicates the source interface name.	`DF0/1`	`Device(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`	`Device(config)#` **`cns event ce-address 11025`**

How to Configure DHCP Zero Touch

Enabling Cisco Networking Service to Receive DHCP Option 43 Messages

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**cns dhcp** **Example:** `Device(config)#` **cns dhcp**	Enables Cisco Networking Service with permission to process the incoming DHCP Option 43 message.
Step 4	**exit** **Example:** `Device# exit`	Exits global configuration mode.

Configuration Examples for DHCP Zero Touch

Example: Using DHCP Option 43 to Retrieve the Initial Configuration File

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.

```
Device(config)# cns event 10.10.10.1
Device(config)# cns config partial 10.10.10.1 inventory
Device(config)# cns exec
Device(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

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WSMA commands	Cisco IOS Web Services Management Agent Command Reference
IP access lists	*Security Configuration Guide: Access Control Lists* in the *Securing the Data Plan Configuration Guide Library*
IP access lists commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*
Public Key Infrastructure	*Public Key Infrastructure Configuration Guide* in the *Secure Connectivity Configuration Guide Library*
Secure Shell and Secure Shell Version 2	*Secure Shell Configuration Guide* in the *Securing User Services Configuration Guide Library*

Related Topic	Document Title
Security commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*
WSMA schema files in XSD format	ftp://ftp.cisco.com/pub/wsma/schema/

RFCs

RFC	Title
RFC 2132	*DHCP Options and BOOTP Vendor Extensions*
RFC 2246	*The TLS Protocol Version 1.0*
RFC 4251	*The Secure Shell (SSH) Protocol Architecture*
RFC 4252	*The Secure Shell (SSH) Authentication Protocol*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for DHCP Zero Touch

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 DHCP Zero Touch

Feature Name	Releases	Feature Information
DHCP Zero Touch	Cisco IOS XE Release 3.8S Cisco IOS XE Release 3.3SE	The DHCP Zero Touch feature allows you to configure the attributes of a device at initial deployment from a DHCP server. DHCP option 43 allows hands-free zero touch deployments. The following commands were introduced or modified: **wsma dhcp**, **cns dhcp**.