NETCONF over BEEP

First Published: June 19, 2006
Last Updated: June 19, 2006

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).

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.

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.

NETCONF over BEEP sends 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.

Finding Feature Information in This Module

Your Cisco IOS software release may not support all of the features documented in this module. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for NETCONF over BEEP" section on page 21.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

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

CISCO SYSTEMS

Corporate Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Contents

Prerequisites for NETCONF over BEEP

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

Restrictions for NETCONF over BEEP

You must be running a crypto image in order configure BEEP using TLS.

Information About NETCONF over BEEP

To configure the NETCONF over BEEP feature, you should understand the following concepts:

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 NATs.

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.

NETCONF Notifications

NETCONF sends notifications of any configuration change over NETCONF if the NETCONF peer has requested notifications to be sent. 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 in the configuration that is changed.

Simple Authentication and Security Layer

The Simple Authentication and Security Layer (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 Transport Layer Security (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 BEEP 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 with NETCONF 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 NETCONF.

For more information about configuring access lists, see the "IP Access Lists" section of the *Cisco IOS IP Application Services Configuration Guide*, Release 12.4.

How to Configure NETCONF over BEEP

This section contains the following tasks:

Configuring a SASL Profile

To enable NETCONF over BEEP using SASL, you must first configure a SASL profile, which specifies which users are allowed access into the router.

Perform this task to configure a SASL Profile.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sasl profile** *profile-name*
4. **mechanism digest-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.
Step 3	**sasl profile** *profile-name* **Example:** `Router(config)# sasl profile beep`	Configures a SASL profile and enters SASL profile configuration mode.
Step 4	**mechanism digest-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 a SASL server.

Enabling NETCONF over BEEP

Perform this task to enable NETCONF over BEEP.

Prerequisites

- There must 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 a SASL profile.

Restrictions

- 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 Rivest, Shamir, and Adelman (RSA) key pairs and specifies that the general-purpose key pair should be generated. Perform this step only once.
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).

	Command or Action	Purpose
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 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(s) for your router from the certificate authority (CA).
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.

	Command or Action	Purpose
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

Notifications are a Cisco extension to the NETCONF standard and are sent only if the NETCONF peer requests notifications to be sent.

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="netconf.8.0"><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="netconf.8.12"><notification-off/></rpc>
```

Use the following XML 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:element name="xml-oper-data">
      <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 xml format.</xs:documentation>
         </xs:annotation>
         <xs:sequence>
            <xs:any/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="xml-config-data">
      <xs:complexType>
         <xs:annotation>
            <xs:documentation> This element is included in the response to get-config and
get operations. Content of this element is the configuration data in xml
format.</xs:documentation>
         </xs:annotation>
         <xs:sequence>
            <xs:any/>
         </xs:sequence>
      </xs:complexType>
```

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 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 command-line interface (CLI), CLI block, and XML format:

NETCONF <edit-config> Request: CLI Format

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <edit-config>
      <target>
```

```
         <running/>
      </target>
      <config>
         <cpi:config-format-xml xmlns="http://www.cisco.com/edi_20/Cat3550/12.1">
            <Hostname>test</Hostname>
            <Interface>
               <InterfaceName>fastEthernet0/1</InterfaceName>
               <IP>
                  <Address>
                     <IPAddress>192.168.1.1</IPAddress>
                     <Mask>255.255.255.0</Mask>
                  </Address>
               </IP>
            </Interface>
         </cpi:config-format-xml>
      </config>
   </edit-config>
</rpc>
```

NETCONF <edit-config> Response: CLI Format

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

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

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cpi:config-format-ios-text-block>
            hostname test
            interface fastEthernet0/1
            ip address 192.168.1.1 255.255.255.0
         </cpi:config-format-ios-text-block>
      </config>
   </edit-config>
</rpc>
```

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

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

NETCONF <edit-config> Request: XML Format

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cpi:config-format-xml xmlns="http://www.cisco.com/edi_20/Cat3550/12.1">
            <Hostname>test</Hostname>
            <Interface>
               <InterfaceName>fastEthernet0/1</InterfaceName>
               <IP>
                  <Address>
```

```
                     <IPAddress>192.168.1.1</IPAddress>
                     <Mask>255.255.255.0</Mask>
                  </Address>
               </IP>
            </Interface>
         </cpi:config-format-xml>
      </config>
   </edit-config>
</rpc>
```

NETCONF <edit-config> Response: XML Format

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

NETCONF uses the <get-config> function to retrieve all or part of a configuration. The <source> element is the name of the configuration database being queried. The <filter> element identifies the portions of the device configuration to retrieve. If the <filter> element is empty or unspecified, the entire configuration is returned.

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

NETCONF <get-config> Request: CLI Format

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <cpi:config-format-ios-text-cmd>
            <cpi:ios-text-filter-spec> | interface </cpi:ios-text-filter-spec>
         </cpi:config-format-ios-text-cmd>
</filter>
   </get-config>
</rpc>
```

NETCONF <get-config> Response: CLI Format

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <data>
      <cpi:cli-data>
         <cpi:cmd>interface fastEthernet0/1</cpi:cmd>
         <cpi:cmd>interface fastEthernet0/2</cpi:cmd>
      </cpi:cli-data>
   </data>
</rpc-reply>
```

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

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <cpi:config-format-ios-text-block>
            <cpi:ios-text-filter-spec> | interface </cpi:ios-text-filter-spec>
         </cpi:config-format-ios-text-block>
      </filter>
```

```
   </get-config>
</rpc>
```

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

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
   <data>
      <cpi:cli-data-block>
         interface fastEthernet0/1
         interface fastEthernet0/2
      </cpi:cli-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

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
    <get>
       <filter>
          <cpi:config-format-cli-ios-text-cmd>
             <ios-filter-text-spec> | include interface </ios-filter-text-spec>
          </cpi:config-format-cli-ios-text-cmd>
          <cpi:cli-operational-data>
             <show>interfaces</show>
             <show>arp</show>
          </cpi:cli-operational-data>
       </filter>
    </get>
 </rpc>
```

NETCONF <get> Response: CLI Format

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
    <data>
       <cpi:cli-data>
          <cpi:cmd>interface fastethernet0/1</cpi:cmd>
          <cpi:cmd>interface loopback</cpi:cmd>
          <cpi:cmd>interface gigabit</cpi:cmd>
       </cpi:cli-data>
       <cpi:cli-operational-data>
          <item>
             <show>interfaces</show>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          <show>arp</show>
          <item>
             <show>arp</show>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
```

```
      </cpi:cli-operational-data>
   </data>
</rpc-reply>
```

NETCONF <get> Request: CLI-Block Format

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
    <get>
       <filter>
          <cpi:config-format-cli-ios-text-block>
             <ios-filter-text-spec> | include interface </ios-filter-text-spec>
          </cpi:config-format-cli-ios-text-block>
          <cpi:cli-operational-data>
             <show>interfaces</show>
             <show>arp</show>
          </cpi:cli-operational-data>
       </filter>
    </get>
 </rpc>
```

NETCONF <get> Response: CLI-Block Format

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema">
    <data>
       <cpi:cli-data-block>
          interface fastethernet0/1
          interface loopback
          interface gigabit
       </cpi:cli-data-block>
       <cpi:cli-operational-data>
          <item>
             <show>interfaces</show>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          <show>arp</show>
          <item>
             <show>arp</show>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cpi:cli-operational-data>
    </data>
</rpc-reply>
```

Monitoring and Maintaining NETCONF Sessions

Perform this task to monitor and maintain NETCONF sessions.

Restrictions

- 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. **show netconf** {**counters** | **session**}
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.
Step 2	**show netconf** {**counters**	**session**} **Example:** `Router# show netconf counters`	Clears NETCONF statistics counters and NETCONF sessions and frees associated resources and locks.
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 over BEEP

This section provides the following configuration example:

- Configuring NETCONF over BEEP: Example, page 15

Configuring NETCONF over BEEP: Example

The following example shows how to configure NETCONF over BEEP:

```
configure terminal

crypto key generate rsa general-keys
crypto pki trustpoint my_trustpoint
enrollment url http://10.2.3.3:80
subject-name CN=dns_name_of_host.com
revocation-check none

crypto pki authenticate my_trustpoint
crypto pki enroll my_trustpoint
line vty 0 15
```

```
netconf lock-time 60
netconf max-sessions 16

netconf beep initiator host1 23 user my_user password my_password encrypt my_trustpoint
reconnect-time 60

netconf beep listener 23 sasl user1 encrypt my_trustpoint
```

Additional References

The following sections provide references related to NETCONF over BEEP.

Related Documents

Related Topic	Document Title
NETCONF over SSHv2	*NETCONF over SSHv2*
Secure Shell and Secure Shell Version 2	"Configuring Secure Shell" and "Configuring Secure Shell Version 2 Support" sections of "Part 6: Other Security Features" in the *Cisco IOS Security Configuration Guide*, Release 12.4.
Security Commands	*Cisco IOS Security Command Reference*, Release 12.4.
IP Access Lists	"Configuring IP Access Lists" section of the *Cisco IOS IP Application Services Configuration Guide*, Release 12.4.
IP Access Lists Commands	*Cisco IOS IP Application Services Command Reference*, Release 12.4 T.

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
2222	*Simple Authentication and Security Layer (SASL)*
2246	*The TLS Protocol Version 1.0*
3080	*The Blocks Extensible Exchange Protocol Core*

Technical Assistance

Description	Link
The Cisco Technical Support & Documentation website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Command Reference

This section documents new commands.

- **netconf beep initiator**
- **netconf beep listener**

netconf beep initiator

To configure Blocks Extensible Exchange Protocol (BEEP) as the transport protocol for Network Configuration Protocol (NETCONF), and to configure a peer as the BEEP initiator, use the **netconf beep initiator** command in global configuration mode. To disable the BEEP initiator, use the **no** form of this command.

netconf beep initiator {*hostname* | *ip-address*} *port-number* **user** *sasl-user* **password** *sasl-password* [**encrypt** *trustpoint*] [**reconnect-time** *seconds*]

no netconf beep initiator {*hostname* | *ip-address*} *port-number*

Syntax Description

hostname	Hostname of the remote device.
ip-address	IP address of the remote device.
port-number	Specifies the BEEP port to use. The valid range is 1 to 65535.
user *sasl-user*	Specifies the SASL user on the far end for this NETCONF session.
password *sasl-password*	Sets the password for the SASL user on the far end.
encrypt *trustpoint*	(Optional) Configures transport layer security (TLS) on this NETCONF session.
reconnect-time *seconds*	(Optional) Specifies the retry timeout for the NETCONF session. The range is 3 to 3600 seconds.

Command Default

BEEP is not enabled as the transport protocol for NETCONF sessions.

Command Modes

Global configuration

Command History

Release	Modification
12.4(9)T	This command was introduced.

Usage Guidelines

Use the **netconf beep initiator** command to specify BEEP as the transport protocol for NETCONF sessions and to specify a peer as the BEEP listener. BEEP is a peer-to-peer client-server protocol. Each peer is labeled in the context of the role it plays 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; similarly, 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.

Use the optional **encrypt** keyword to configure BEEP to use transport layer security (TLS) to provide simple security for NETCONF sessions.

Examples

The following example shows how to enable NETCONF over BEEP and to configure a BEEP peer as the BEEP initiator:

```
netconf beep initiator host1 25 user user1 password password1 encrypt 23 reconnect-time 60
```

Related Commands

Command	Description
netconf beep listener	Configures BEEP as the transport protocol for NETCONF and configures a peer as the BEEP listener.

netconf beep listener

To configure Blocks Extensible Exchange Protocol (BEEP) as the transport protocol for Network Configuration Protocol (NETCONF), and to configure a peer as the BEEP listener, use the **netconf beep listener** command in global configuration mode. To disable the BEEP listener, use the **no** form of this command.

netconf beep listener [*port-number*] [**acl** *access-list-number*] [**sasl** *sasl-profile*] [**encrypt** *trustpoint*]

no netconf beep listener

Syntax Description

port-number	(Optional) Specifies which BEEP port on which to listen.
acl *access-list-number*	(Optional) Specifies the access-control list to be applied to restrict incoming client connections.
sasl *sasl-profile*	(Optional) Configures a Simple Authentication and Security Layer (SASL) profile to use during session establishment.
encrypt *trustpoint*	(Optional) Configures transport layer security (TLS) on a NETCONF session.

Command Default

BEEP is not enabled as the transport protocol for NETCONF sessions.

Command Modes

Global configuration

Command History

Release	Modification
12.4(9)T	This command was introduced.

Usage Guidelines

Use the **netconf beep listener** command to specify BEEP as the transport protocol for NETCONF sessions and to specify a peer as the BEEP listener.

BEEP is a peer-to-peer client-server protocol. Each peer is labeled in the context of the role it plays 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; similarly, 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.

You must configure a SASL profile before you can configure NETCONF over BEEP to use SASL during session establishment.

Examples

The following example shows how to configure NETCONF over BEEP and to specify a peer as the BEEP listener:

```
sasl profile beep
 mechanism digest-md5
 server user user1 password password1
 exit
netconf beep listener 23 acl 1 sasl beep encrypt 25
```

Related Commands

Command	Description
netconf beep initiator	Configures BEEP as the transport protocol for NETCONF and configures a peer as the BEEP initiator.

Feature Information for NETCONF over BEEP

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for NETCONF over SSHv2

Feature Name	Releases	Feature Information
NETCONF over BEEP	12.4(9)T	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). 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. • In 12.4(9)T, this feature was introduced.