

System Security Configuration Guide for Cisco NCS 560 Series Routers, IOS XR Release 26.1.x, 26.2.x, 26.3.x

First Published: 2026-01-29

Last Modified: 2026-09-11

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, 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.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

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

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2026 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1 **New and Changed Feature Information** 1

System Security Features Added or Modified in IOS XR Release 26.x.x 1

CHAPTER 2 **YANG Data Models for System Security Features** 3

Using YANG Data Models 3

CHAPTER 3 **Configuring AAA Services** 5

Configuring AAA Services 5

Prerequisites for Configuring AAA Services 5

Restrictions for Configuring AAA Services 6

Configure Task group 6

Configure User Groups 8

Configure First User on Cisco Routers 9

Configure Users 10

Password Masking For Type 7 Password Authentication 12

Configure Type 8 and Type 9 Passwords 13

Configure Type 10 Password 14

Backward Compatibility for Password Types 16

Configure AAA Password Policy 16

Configure Password Policy for User Secret and Password 18

Password Policy to Restrict Consecutive Characters 21

Deprecation of Type 7 password and Type 5 secret 25

Configure Router to RADIUS Server Communication 31

Configure RADIUS Dead-Server Detection 34

Configure TACACS+ Server 36

Configure RADIUS Server Groups 38

Configure TACACS+ Server Groups 40
Configure Per VRF TACACS+ Server Groups 42
TACACS+ with TLS protection 43
Create Series of Authentication Methods 47
Create Series of Authorization Methods 48
Create Series of Accounting Methods 50
Generate Interim Accounting Records 52
Apply Method List 53
Enable Accounting Services 54
Configure Login Parameters 55
Task Maps 56
How to Configure Hold-Down Timer for TACACS+ 58
Overview on AAA Services 61
RADIUS with TLS protection 81
Hold-Down Timer for TACACS+ 86
Model-based AAA 88
Command Authorization Using Local User Account 97

CHAPTER 4 **Implementing Certification Authority Interoperability** 103
Implementing Certification Authority Interoperability 103
Prerequisites for Implementing Certification Authority 103
Restrictions for Implementing Certification Authority 104
Configure Router Hostname and IP Domain Name 105
RSA key pairs 106
Generate RSA Key Pair 107
Import Public Key to the Router 108
Declare Certification Authority and Configure Trusted Point 109
Authenticate CA 111
Request Your Own Certificates 111
Configure Certificate Enrollment Using Cut-and-Paste 112
Certificate Authority Trust Pool Management 115
CA Certificate Bundling in the Trust Pool 116
Updating the CA Trustpool 116
Configuring Optional Trustpool Policy Parameters 118

Handling of CA Certificates appearing both in Trust Pool and Trust Point 118
Expiry Notification for PKI Certificate 118
Learn About the PKI Alert Notification 119
Enable PKI Traps 120
Regenerate the Certificate 120
Integrating Cisco IOS XR and Crosswork Trust Insights 121
How to Integrate Cisco IOS XR and Crosswork Trust Insights 122
Generate Key Pair 124
Generate System Trust Point for the Leaf and Root Certificate 125
Generate Root and Leaf Certificates 126
System Certificates Expiry 128
Collect Data Dossier 129
Procedure to Test Key Generation and Data-signing with Different Key Algorithm 132
Verify Authenticity of RPM Packages Using Fingerprint 132
Support for Ed25519 Public-Key Signature System 135
Generate Crypto Key for Ed25519 Signature Algorithm 135
Information About Implementing Certification Authority 136
Supported Standards for Certification Authority Interoperability 136
Certification Authorities 137

CHAPTER 5 **Implementing Keychain Management** 139
Implementing Keychain Management 139
Restrictions for Implementing Keychain Management 139
Configure Keychain 139
Configure Tolerance Specification to Accept Keys 141
Configure Key Identifier for Keychain 141
Configure Text for Key String 142
Determine Valid Keys 143
Configure Keys to Generate Authentication Digest for Outbound Application Traffic 144
Configure Cryptographic Algorithm 145
Lifetime of Key 147

CHAPTER 6 **Implementing Type 6 Password Encryption** 149
How to Implement Type 6 Password Encryption 149

Enabling Type6 Feature and Creating a Primary Key (Type 6 Server) 149
Implementing Key Chain for BGP Sessions (Type 6 Client) 152
Creating a BGP Session (Type 6 Password Encryption Use Case) 153

CHAPTER 7 802.1X Port-Based Authentication 155
Usage guidelines and restrictions for 802.1X port-based authentication 156
IEEE 802.1X Device Roles 156
Understanding 802.1X Port-Based Authentication 157
802.1X host-modes 158
Configure 802.1X host-modes 158
Prerequisites for 802.1X Port-Based Authentication 158
802.1X with Remote RADIUS Authentication 159
Configure RADIUS Server 159
Configure 802.1X Authentication Method 159
Configure 802.1X Authenticator Profile 159
Configure 8021X Profile on Interface 160
802.1X with Local EAP Authentication 161
Generate RSA Key Pair 161
Configure Trustpoint 161
Configure Domain Name 162
Certificate Configurations 162
Configure EAP Profile 163
Configure 802.1X Authenticator Profile 163
Configure 802.1X Profile on Interface 164
Router as 802.1X Supplicant 164
Configure 802.1X Supplicant Profile 164
Configure 802.1X Profile on Interface 165
Verify 802.1X Port-Based Authentication 165
Show Command Outputs 165
Syslog Messages 166

CHAPTER 8 Implementing MAC Authentication Bypass 169
MAC Authentication Bypass 170
Restrictions for MAC Authentication Bypass 170

Authentication Failure Scenarios of MAB 171
How MAC Authentication Bypass Works 172
Configure MAC Authentication Bypass 173
Verify MAC Authentication Bypass 175
System Logs for MAC Authentication Bypass 177

CHAPTER 9 **Understanding URPF 179**
Configuring uRPF Loose Mode 179
uRPF strict mode with external TCAM 180
Restrictions for uRPF strict mode with external TCAM 181
Guidelines for uRPF strict mode with external TCAM 181
Configure uRPF strict mode 181

CHAPTER 10 **Implementing Management Plane Protection 183**
Benefits of Management Plane Protection 183
Restrictions for Implementing Management Plane Protection 184
Configure Device for Management Plane Protection for Inband Interface 184
Configure Device for Management Plane Protection for Out-of-band Interface 187
Information About Implementing Management Plane Protection 191
Peer-Filtering on Interfaces 191
Control Plane Protection 191
Management Plane 191

CHAPTER 11 **Traffic Protection for Third-Party Applications 193**
Example: Traffic Protection for Third-Party Applications over gRPC 193
Limitations for Traffic Protection for Third-Party Applications over gRPC 194
Prerequisites for Traffic Protection for Third-Party Applications vver gRPC 194
Configuring Traffic Protection for Third-Party Applications 194
Troubleshooting Traffic Protection for Third-Party Applications 195

CHAPTER 12 **Implementing Secure Shell 197**
Implementing Secure Shell 197
Information About Implementing Secure Shell 197
SSH Server 197

SSH Client 198
SFTP Feature Overview 199
RSA Based Host Authentication 200
RSA Based User Authentication 200
SSHv2 Client Keyboard-Interactive Authentication 201
Prerequisites for Implementing Secure Shell 201
SSH and SFTP in Baseline Cisco IOS XR Software Image 202
FIDO2 authentication for SSH 202
Restrictions for configuring FIDO2 authentication for SSH 203
Guidelines for configuring FIDO support for SSH 203
Configure FIDO2 authentication for SSH 203
Netconf access controls 205
Benefits of Netconf access control 205
Best practice for Netconf access control 205
Restrictions for Netconf access control 205
How Netconf access control works 206
Configure Netconf access control 206
Guidelines and Restrictions for Implementing Secure Shell 207
Configure SSH 208
SSH client strict host key checks 212
Restrictions for SSH client strict host key check 212
Configure SSH client strict host key checking 213
SSH client knownhosts commands 215
Automatic generation of SSH host-key pairs 215
Configure the Allowed SSH Host-Key Pair Algorithms 216
Ed25519 Public-Key Signature Algorithm Support for SSH 218
How to Generate Ed25519 Public Key for SSH 219
Configure SSH Client 219
Order of SSH Client Authentication Methods 221
How to Set the Order of Authentication Methods for SSH Clients 222
SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm 222
Disable HMAC Algorithm 223
Enable Cipher Public Key 223
User Configurable Maximum Authentication Attempts for SSH 225

Configure Maximum Authentication Attempts for SSH 226
X.509v3 Certificate-based Authentication for SSH 227
Configure X.509v3 Certificate-based Authentication for SSH 230
SSH Port Forwarding 235
How to Enable SSH Port Forwarding 236
Non-Default SSH Port 238
How to Configure Non-Default SSH Port 239
Multi-Factor Authentication for SSH 242
Multi-Factor Authentication Workflow 243
Set Up Multi-Factor Authentication for SSH 244
Configure Duo System for MFA 244
Configure Duo Authentication Proxy for MFA 245
Configure ISE for MFA 245
Configure RADIUS Server Attributes for MFA 246
Verify MFA Set-up for SSH Connection 246
SSH server timeouts 247
Unused connection timeout for SSH sessions 247
Set unused connection timeout for SSH sessions 248
Channel timeout for SSH sessions 251
Set channel timeout for SSH sessions 252

CHAPTER 13 **Gated shell access** 255
Key features of gated shell access 255
How gated shell access works 256
Restrict shell access using a Cisco-signed Consent Token 257
Unrestrict shell access using a Cisco-signed Consent Token 258
Restrict shell access using an owner Consent Token 259
Unrestrict shell access using an owner Consent Token 260
Terminate a pending Consent Token authorization request 261
Access a restricted shell 261

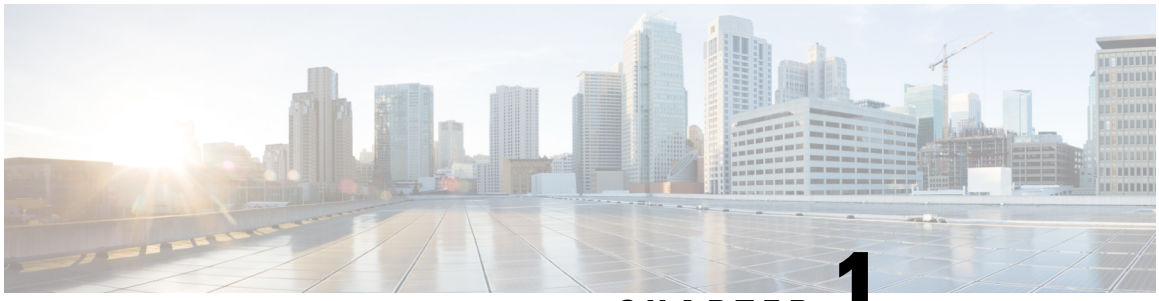
CHAPTER 14 **Implementing Lawful Intercept** 263
Interception Mode 263
Data Interception 264

Lawful Intercept Topology 264
Benefits of Lawful Intercept 265
Information About Lawful Intercept Implementation 265
Prerequisites for Implementing Lawful Intercept 265
Installing Lawful Intercept (LI) Package 266
Installing and Activating the LI Package 266
Deactivating the LI RPM 267
Types of Lawful Intercept Mediation Device 267
Restrictions for Implementing Lawful Intercept 267
Limitations of Lawful Intercept 269
Scale or Performance Values 269
How to Configure SNMPv3 Access for Lawful Intercept 269
Disabling SNMP-based Lawful Intercept 269
Configuring the Inband Management Plane Protection Feature 270
Enabling the Lawful Intercept SNMP Server Configuration 270
Additional Information on Lawful Intercept 271
Intercepting IPv4 and IPv6 Packets 271
Lawful Intercept Filters 271
Encapsulation Type Supported for Intercepted Packets 271
High Availability for Lawful Intercept 272
Preserving TAP and MD Tables during RP Fail Over 272
Replay Timer 272

CHAPTER 15 **Cisco MASA Service** 273
Why Do I Need Cisco MASA? 274
Use Cases for Ownership Vouchers 274
Authentication Flow 275
Interacting with the MASA Server 277
Interacting with MASA Through Web Application 279
Interacting with MASA Through REST APIs 282
Interaction with MASA through gRPC 283
Workflow to Provision a Router Using Ownership Voucher 284

CHAPTER 16 **EST protocol for Automated Certificate Provisioning** 287

EST protocol for automated certificate provisioning 287
Configure the EST protocol 292



CHAPTER 1

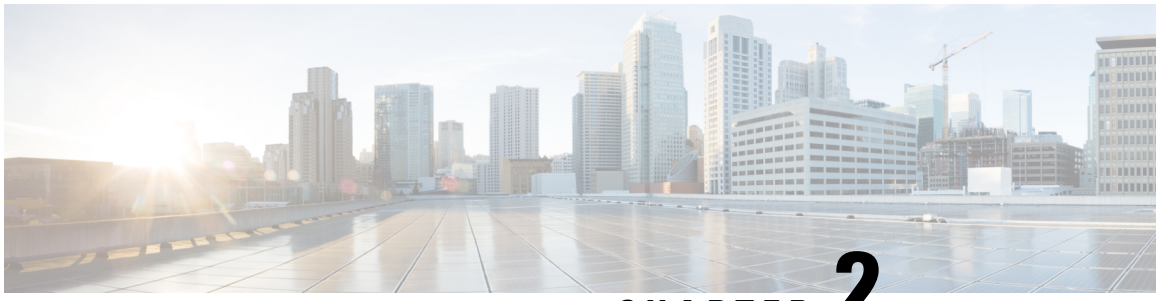
New and Changed Feature Information

This table summarizes the new and changed feature information for the *System Security Configuration Guide*, and tells you where they are documented.

- System Security Features Added or Modified in IOS XR Release 26.x.x, on page 1

System Security Features Added or Modified in IOS XR Release 26.x.x

Feature	Description	Changed in Release	Where Documented
Gated shell access	This feature was introduced.	Release 26.3.1	Gated shell access, on page 255
SSH client strict host key check	This feature was introduced.	Release 26.3.1	SSH client strict host key checks, on page 212
uRPF strict mode	This feature was introduced.	Release 26.2.1	uRPF strict mode with external TCAM, on page 180
SSH key strength: 3072 bits by default	This feature was introduced.	Release 26.1.1	Automatic generation of SSH host-key pairs , on page 215



CHAPTER 2

YANG Data Models for System Security Features

This chapter provides information about the YANG data models for System Security features.

• Using YANG Data Models, on page 3

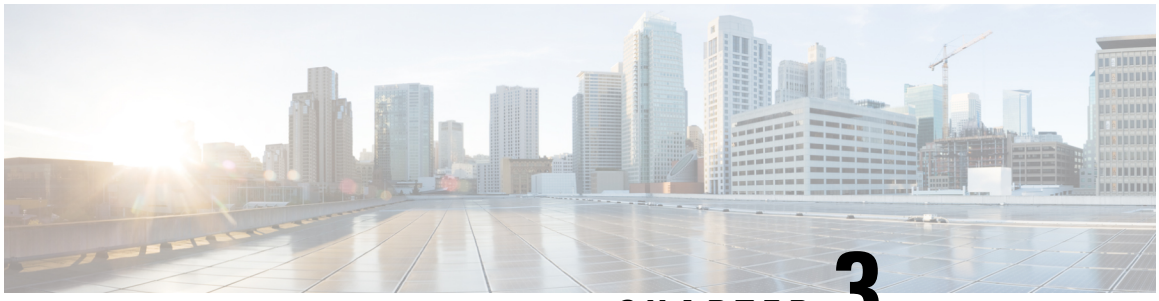
Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the **Available-Content.md** file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.



CHAPTER 3

Configuring AAA Services

This module describes the implementation of the administrative model of *task-based authorization* used to control user access in the software system. The major tasks required to implement task-based authorization involve configuring user groups and task groups.

User groups and task groups are configured through the software command set used for authentication, authorization and accounting (AAA) services. Authentication commands are used to verify the identity of a user or principal. Authorization commands are used to verify that an authenticated user (or principal) is granted permission to perform a specific task. Accounting commands are used for logging of sessions and to create an audit trail by recording certain user- or system-generated actions.

AAA is part of the software base package and is available by default.

- Configuring AAA Services, on page 5

Configuring AAA Services

This module describes the implementation of the administrative model of *task-based authorization* used to control user access in the software system. The major tasks required to implement task-based authorization involve configuring user groups and task groups.

User groups and task groups are configured through the software command set used for authentication, authorization and accounting (AAA) services. Authentication commands are used to verify the identity of a user or principal. Authorization commands are used to verify that an authenticated user (or principal) is granted permission to perform a specific task. Accounting commands are used for logging of sessions and to create an audit trail by recording certain user- or system-generated actions.

AAA is part of the software base package and is available by default.

Prerequisites for Configuring AAA Services

The following are the prerequisites to configure AAA services:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- Establish a root system user using the initial setup dialog. The administrator may configure a few local users without any specific AAA configuration. The external security server becomes necessary when user accounts are shared among many routers within an administrative domain. A typical configuration

would include the use of an external AAA security server and database with the local database option as a backup in case the external server becomes unreachable.

Restrictions for Configuring AAA Services

This section lists the restrictions for configuring AAA services.

Compatibility

Compatibility is verified with the Cisco freeware TACACS+ server and FreeRADIUS only.

Interoperability

Router administrators can use the same AAA server software and database (for example, CiscoSecure ACS) for the router and any other Cisco equipment that does not currently run the Cisco software. To support interoperability between the router and external TACACS+ servers that do not support task IDs, see the "Task IDs for TACACS+ and RADIUS Authenticated Users, on page 77" section.

Character Policy

These are the characters which are not allowed for **aaa group server tacacs+**.

- Backtick: `
- Length > 40 characters
- Unescaped space or parentheses in an unquoted name

Configure Task group

Task-based authorization employs the concept of a *task ID* as its basic element. A task ID defines the permission to execute an operation for a given user. Each user is associated with a set of permitted router operation tasks identified by task IDs. Users are granted authority by being assigned to user groups that are in turn associated with task groups. Each task group is associated with one or more task IDs. The first configuration task in setting up an authorization scheme to configure the task groups, followed by user groups, followed by individual users.

Specific task IDs can be removed from a task group by specifying the **no** prefix for the **task** command.

The task group itself can be removed. Deleting a task group that is still referred to elsewhere results in an error.

Before you begin

Before creating task groups and associating them with task IDs, you should have some familiarity with the router list of task IDs and the purpose of each task ID. Use the **show aaa task supported** command to display a complete list of task IDs.

Note Only users with write permissions for the AAA task ID can configure task groups.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **taskgroup** *taskgroup-name*

Example:

```
RP/0/RP0/CPU0:router(config)# taskgroup beta
```

Creates a name for a particular task group and enters task group configuration submode.

- Specific task groups can be removed from the system by specifying the **no** form of the **taskgroup** command.

Step 3 **description** *string*

Example:

```
RP/0/RP0/CPU0:router(config-tg)# description this is a sample task group description
```

(Optional) Creates a description of the task group named in Step 2.

Step 4 **task** {**read** | **write** | **execute** | **debug**} *taskid-name*

Example:

```
RP/0/RP0/CPU0:router(config-tg)# task read bgp
```

Specifies a task ID to be associated with the task group named in Step 2.

- Assigns **read** permission for any CLI or API invocations associated with that task ID and performed by a member of the task group.
- Specific task IDs can be removed from a task group by specifying the **no** prefix for the **task** command.

Step 5 Repeat for each task ID to be associated with the task group named in Step 2.

—

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

After completing configuration of a full set of task groups, configure a full set of user groups as described in the Configuring User Groups section.

Configure User Groups

User groups are configured with the command parameters for a set of users, such as task groups. Entering the **usergroup** command accesses the user group configuration submode. Users can remove specific user groups by using the **no** form of the **usergroup** command. Deleting a usergroup that is still referenced in the system results in a warning.

Before you begin

Note Only users associated with the WRITE:AAA task ID can configure user groups. User groups cannot inherit properties from predefined groups, such as owner-sdr.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **usergroup** *usergroup-name*

Example:

```
 RP/0/RP0/CPU0:router(config)# usergroup beta
```

Creates a name for a particular user group and enters user group configuration submode.

- Specific user groups can be removed from the system by specifying the **no** form of the **usergroup** command.

Step 3 **description** *string*

Example:

```
 RP/0/RP0/CPU0:router(config-ug)#
description this is a sample user group description
```

(Optional) Creates a description of the user group named in Step 2.

Step 4 **inherit usergroup** *usergroup-name*

Example:

```
 RP/0/RP0/CPU0:router(config-ug)#
inherit usergroup sales
```

- Explicitly defines permissions for the user group.

Step 5 **taskgroup** *taskgroup-name*

Example:

```
RP/0/RP0/CPU0:router(config-ug)# taskgroup beta
```

Associates the user group named in Step 2 with the task group named in this step.

- The user group takes on the configuration attributes (task ID list and permissions) already defined for the entered task group.

Step 6 Repeat Step for each task group to be associated with the user group named in Step 2.

—

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure First User on Cisco Routers

When a Cisco Router is booted for the very first time, and a user logs in for the first time, a root-system username and password must be created. Configure the root-system username and password, as described in the following procedure:

Step 1. Establish a connection to the Console port.

This initiates communication with the router. When you have successfully connected to the router through the Console port, the router displays the prompt:

```
Enter root-system username
```

Step 2. Type the username for the root-system login and press **Enter**.

Sets the root-system username, which is used to log in to the router.

Step 3. Type the password for the root-system login and press **Enter**.

Creates an encrypted password for the root-system username. This password must be at least six characters in length. The router displays the prompt:

```
Enter secret
```

Step 4. Retype the password for the root-system login and press **Enter**.

Allows the router to verify that you have entered the same password both times. The router displays the prompt:

```
Enter secret again
```

Note If the passwords do not match, the router prompts you to repeat the process.

Step 5. Log in to the router.

Establishes your access rights for the router management session.

Note In case of Router reload, when there is no stored username and password, you must create a new username and password.

For more information on minimum password length, see Minimum Password Length for First User Creation, on page 76.

Example

The following example shows the root-system username and password configuration for a new router, and it shows the initial login:

```
/* Administrative User Dialog */
Enter root-system username: cisco
Enter secret:
Enter secret again:

RP/0/0/CPU0:Jan 10 12:50:53.105 : exec[65652]: %MGBL-CONFIG-6-DB_COMMIT : 'Administration
configuration committed by system'.
Use 'show configuration commit changes 2000000009' to view the changes. Use the 'admin'
mode 'configure' command to modify this configuration.

/* User Access Verification */
Username: cisco
Password:
RP/0/0/CPU0:ios#
```

The secret line in the configuration command script shows that the password is encrypted. When you type the password during configuration and login, the password is hidden.

Configure Users

Perform this task to configure a user.

Each user is identified by a username that is unique across the administrative domain. Each user should be made a member of at least one user group. Deleting a user group may orphan the users associated with that group. The AAA server authenticates orphaned users but most commands are not authorized.

From Cisco IOS XR Software Release 24.3.1 and later, the router synchronizes up to 100 valid Linux-compatible users to the Linux infrastructure (/etc/passwd file), and up to 20 users to the standby route processor (RP) in a dual-RP router setup.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **username** *user-name*

Example:

```
RP/0/RP0/CPU0:router(config)# username user1
```

Creates a name for a new user (or identifies a current user) and enters username configuration submode.

- The *user-name* argument can be only one word. Spaces and quotation marks are not allowed.

Step 3 Do one of the following:

- **password** {**0** | **7**} *password*
- **secret** {**0** | **5**} *secret*

Example:

```
RP/0/RP0/CPU0:router(config-un)# password 0 pwd1
```

or

```
RP/0/RP0/CPU0:router(config-un)# secret 0 sec1
```

Specifies a password for the user named in step 2.

- Use the **secret** command to create a secure login password for the user names specified in step 2.
- Entering **0** following the **password** command specifies that an unencrypted (clear-text) password follows. Entering **7, 8, 9, 10** following the **password** command specifies that an encrypted password follows.
- Entering **0** following the **secret** command specifies that a secure unencrypted (clear-text) password follows. Entering **5** following the **secret** command specifies that a secure encrypted password follows.
- Type **0** is the default for the **password** and **secret** commands.

Step 4 **group** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config-un)# group sysadmin
```

Assigns the user named in step 2 to a user group that has already been defined through the **usergroup** command.

- The user takes on all attributes of the user group, as defined by that user group's association to various task groups.
- Each user must be assigned to at least one user group. A user may belong to multiple user groups.

Step 5 Repeat step 4 for each user group to be associated with the user specified in step 2.

—

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Password Masking For Type 7 Password Authentication

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Password Masking	Release 7.3.1	With this feature, when you key in a password or secret, it is not displayed on the screen. This enhances security. The feature is enabled by default. The following options are added to the **username** command: • **masked-password** • **masked-secret**

When you key in a password, to ensure that it is not displayed on the screen, use the **masked-password** option. Details:

Use the **username** command as shown below, and enter the password.

The following command contains the username us3, and 0 to specify a cleartext password.

```
Router(config)# username us3 masked-password 0

Enter password:
Re-enter password:

Router(config)#commit
```

View the encrypted password:

```
Router# show run aaa
..

username us3
 password 7 105A1D0D
```

Enable Type 7 password authentication and enter the encrypted password 105A1D0D. You can also use a password encrypted earlier.

```
Router(config)# username us3 masked-password 7

Enter password:
Re-enter password:


Router(config)#commit
```

If there is a password mismatch between the two entries, an error message is displayed.

Configure Type 8 and Type 9 Passwords

When configuring a password, user has the following two options:

- User can provide an already encrypted value, which is stored directly in the system without any further encryption.
- User can provide a cleartext password that is internally encrypted and stored in the system.

The Type 5, Type 8, and Type 9 encryption methods provide the above mentioned options for users to configure their passwords.

For more information about configuring users with Type 8 and Type 9 encryption methods, see Configure Users, on page 10 section.

Configuration Example

Directly configuring a Type 8 encrypted password:

```
Router(config)# username demo8
Router(config-un)#secret 8 $8$dsYGNam3K1SIJO$7nv/35M/qr6t.dVc7UY9zrJDWRVqncHub1PE9UlMQFs
```

Configuring a clear-text password that is encrypted using Type 8 encryption method:

```
Router(config)# username demo8
Router(config-un)#secret 0 enc-type 8 PASSWORD
```

Directly configuring a Type 9 encrypted password:

```
Router(config)# username demo9
Router(config-un)# secret 9 $9$nhEmQVczB7dqsO$X.HsgL6x1il0RxkOSSvyQYwucySCt7qFm4v7pqCxkKM
```

Configuring a clear-text password that is encrypted using Type 9 encryption method:

```
Router(config)# username demo9
Router(config-un)#secret 0 enc-type 9 PASSWORD
```

Password Masking For Type 5, Type 8, Type 9 And Type 10 Password Authentication

When you key in a password, to ensure that it is not displayed on the screen, use the **masked-secret** option. Steps:

Use the **username** command as shown below, and enter the password.

The following command contains the username us6, 0 to specify a cleartext password, and the encryption type (5, 8, 9, or 10).

```
Router(config)# username us6 masked-secret 0 enc-type 8
```

```
Enter secret:
Re-enter secret:

Router(config)# commit
```

View the encrypted secret:

```
Router# show running-config aaa
..
username us6
 secret 8 $8$mlcSk/Ae5Qu/5k$RJdI3SQ8B4iP7rdxxQvVlJVeRHSubZzcgcaLYxjg36s
```

Enter the username, 8 to specify Type 8 secret authentication, and enter the Type 8 secret. You can also use a secret encrypted earlier.

```
Router(config)# username us6 masked-secret 8

Enter secret:
Re-enter secret:

Router(config)# commmit
```

If there is a password mismatch between the two entries, an error message is displayed.

Related Topics

- Type 8 and Type 9 Passwords, on page 72
- Type 10 Password, on page 72

Associated Commands

- secret
- username

Configure Type 10 Password

You can use these options to configure Type 10 password (that uses **SHA512** hashing algorithm) for the user:

Configuration Example

From Release 7.0.1 and later, Type 10 is applied by default for the passwords when you create a user with a clear-text password.

```
Router#configure
Router(config)#username user10 secret testpassword
Router(config-un)#commit
```

Also, a new parameter '10' is available for the **secret** option under the **username** command to configure explicitly the Type 10 passwords.

```
Router#configure
Router(config)#username root secret 10
$6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqLwX8UsSEr9ApG6c5pzhMJmZtgW4jObAQ7meAwyhu5VM/aRFJqe/jxZG17h6xPrvJWf1
Router(config-un)#commit
```

In scenarios where you have to enter the clear-text password, you can specify the encryption algorithm to be used by using the **enc-type** keyword and the clear-text password as follows:

```
Router#configure
Router(config)#username user10 secret 0 enc-type 10 testpassword
Router(config-un)#commit
```

The preceding configuration configures the user with the Type10 password.

In System Admin VM, you can specify the Type 10 encrypted password as follows:

```
Router#admin
sysadmin-vm:0_RP0# configure
sysadmin-vm:0_RP0(config)# aaa authentication users user user10 password testpassword
sysadmin-vm:0_RP0(config)# commit
Commit complete.
sysadmin-vm:0_RP0(config)# end
sysadmin-vm:0_RP0# exit
Router#
```

Running Configuration

```
Router#show running-configuration username user10
!
username user10
secret 10
$6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqLwX8UsSEr9ApG6c5pzhMJmZtgW4jObAQ7meAwyhu5VM/aRFJqe/jxZG17h6xPrvJWf1
!
```

In System Admin VM:

```
sysadmin-vm:0_RP0#show running-configuration aaa authentication users user user10
Tue Jan 14 07:32:44.363 UTC+00:00
aaa authentication users user user10
password
$6$MMvhlj1CzSd2nJfB$Bbzvxzriwx4iLFg75w4zj15YK3yeoq5UoRyc1evtSX0c4EuaMlqK.v7E3zbY1yKKxkN6rXpQuhMJOuyRHItDc1
!
sysadmin-vm:0_RP0#
```

Similarly, you can use the **admin show running-configuration aaa authentication users user user10** command in XR VM, to see the details of the password configured for the user.

Related Topics

- Type 10 Password, on page 72
- Backward Compatibility for Password Types, on page 16

Associated Commands

- secret
- username

Backward Compatibility for Password Types

When you downgrade from Cisco IOS XR Software Release 7.0.1 to lower versions, you might experience issues such as configuration loss, authentication failure, termination of downgrade process or XR VM being down. These issues occur because Type 5 (MD5) is the default encryption for older releases.

It is recommended to follow these steps to avoid such backward compatibility issues during downgrade:

- Perform all install operations for the downgrade except the **install activate** step.
- Before performing the **install activate** step, take the backup of user configurations on both the VMs. You can use the **show running-configuration username | file harddisk:**/*filename* command for the same.
- Delete all users on both the VMs and initiate the **install activate** step.
- When the router boots up with the lower version, it prompts for the first root-system user creation.
- After your login with the credentials of the first user, apply the previously saved configuration to both the VMs.

For example, consider an authentication failure scenario after a downgrade. The downgrade process does not affect any existing user name configuration with Type 5 secret. Such users can log in without any issue using the clear-text password. But, the users with Type 10 configuration might experience authentication failure, and may not be able to log in. In such cases, the system treats the whole string "10<space><sha512-hashed-text>" as a clear-text password and encrypts it to Type 5 (MD5) password. Use that "10<space><sha512-hashed-text> string as the password for that Type 10 user to log in. After you log in with the preceding step, you must explicitly configure the clear-text password in XR VM and System Admin VM as described in the Configuration Example section.

Configure AAA Password Policy

To configure the AAA password policy, use the **aaa password-policy** command in the global configuration mode.

Configuration Example

This example shows how to configure a AAA password security policy, *test-policy*. This *test-policy* is applied to a user by using the **username** command along with **password-policy** option.

```
RP/0/RP0/CPU0:router(config)#aaa password-policy test-policy
RP/0/RP0/CPU0:router(config-aaa)#min-length 8
RP/0/RP0/CPU0:router(config-aaa)#max-length 15
RP/0/RP0/CPU0:router(config-aaa)#lifetime months 3
RP/0/RP0/CPU0:router(config-aaa)#min-char-change 5
RP/0/RP0/CPU0:router(config-aaa)#authen-max-attempts 3
RP/0/RP0/CPU0:router(config-aaa)#lockout-time days 1
RP/0/RP0/CPU0:router(config-aaa)#commit

RP/0/RP0/CPU0:router(config)#username user1 password-policy test-policy password 0 pwd1
```

Running Configuration

```
aaa password-policy test-policy
 min-length 8
```

```
 max-length 15
 lifetime months 3
 min-char-change 5
 authen-max-attempts 3
 lockout-time days 1
 !
```

Verification

Use this command to get details of the AAA password policy configured in the router:

```
RP/0/RP0/CPU0:router#show aaa password-policy

Fri Feb  3 16:50:58.086 EDT
Password Policy Name : test-policy
  Number of Users : 1
  Minimum Length : 8
  Maximum Length : 15
  Special Character Len : 0
  Uppercase Character Len : 0
  Lowercase Character Len : 1
  Numeric Character Len : 0
  Policy Life Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 3
    years : 0
  Lockout Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 1
    months : 0
    years : 0
  Character Change Len : 5
  Maximum Failure Attempts : 3
```

Password Masking For AAA Password Policies

When you key in a password, to ensure that it is not displayed on the screen, use the **masked-password** option.
Steps:

Create a AAA password security policy and enter the cleartext password.

In this example, a policy called *security* is created, and 0 is specified for a cleartext password.

```
Router(config)# aaa password-policy security
Router(config)# username us6 password-policy security masked-password 0

Enter password:
Re-enter password:

Router(config)#commit
```

View the encrypted password:

```
Router# show run aaa
..
```

```
aaa password-policy security
..
username us6
 password-policy security password 7 0835585A
```

Enter the username, 7 to specify Type 7 password authentication, and enter the password 0835585A. You can also use a password encrypted earlier.

```
Router(config)# username us6 password-policy test-policy masked-password 7

Enter password:
Re-enter password:


Router(config)#commit
```

If there is a password mismatch between the two entries, an error message is displayed.

Related Topic

- AAA Password Security for FIPS Compliance, on page 73

Associated Commands

- **aaa password-policy**
- **show aaa password-policy**
- **username**

Configure Password Policy for User Secret and Password

A new option, **policy** is added to the existing **username** command to apply the password policy to the user. This policy is common to the password and the secret. After applying the policy to the user, the system validates any change to the secret or password against that particular policy.

On Cisco IOS XR 64 bit platforms, the first user is synced from XR VM to System Admin VM. If the user is configured for a secret policy, then the password compliance is checked during the configuration. The password is then synced to System Admin VM. When system administrators need to explicitly configure the user, then the username configurations on System Admin VM are not checked for the password compliance. This is because, the password policy configuration is not applicable on System Admin VM.

Note The configuration model for the AAA component on System Admin VM is the YANG file. A change in the YANG model can cause configuration inconsistencies during an upgrade on downgrade scenario.

Guidelines to Configure Password Policy for User Secret

You must follow these guidelines while configuring policy for user password or secret:

- If there is no policy already configured while configuring the user secret, then the system does not have any policy validation to do for that secret. So, you must ensure that the policy is configured first and then applied to the username configuration, before configuring the secret. Especially when you copy and paste the username configurations.

- If you change the user secret at the time of log in, the system applies the same hashing type as it was applied in the username configuration. For example, if the secret was applied as Type 5 in the username configuration, then the system applies Type 5 itself if the secret is modified at the time of log in.
- Password and secret are different entities. Hence, if **restrict-old-count** is configured in the policy while changing the password, the system checks for compliance only with the history of old passwords; not with the history of old secrets.
- Similarly, the system does not check for old password history while changing the secret and conversely. So, if the same secret (in clear text) was used before as password for the user, then the system allows that secret configuration. And, conversely, for the password configuration.
- The **restrict-old-count** applies to both secret and password. So, the configured secret or password overwrites the old secret or password in the FIFO order.
- When you try to assign a different policy to a username which already has a password or secret associated to a policy, then the system rejects that configuration. The error message indicates to remove the existing password or secret in order to apply the new policy to the user.
- The system does not allow any configuration that requires the secret to be validated against the previous composition of the cleartext secret. This is because, you cannot retrieve the clear text format of the secret that was once hashed, for comparison. Hence, the following configurations do not have any effect on the secret configuration of the user:
  - **max-char-repetition**
  - **min-char-change**
  - **restrict-password-reverse**
  - **restrict-password-advanced**
- As the new **policy** configuration for the user is common to password and secret, the existing **password-policy** configuration becomes redundant. So, these configurations must be mutually exclusive. When any one of these configurations is already present, and if you try to configure the other policy, then the system rejects it. The error message says that **password-policy** and **policy** are not allowed together.

Configuration Example

This example shows how to configure a password policy for the user, that applies to both the password and the secret of the user.

```
Router#configure
Router(config)#username user1
Router(config-un)#policy test-policy1
Router(config-un)#secret 10
$6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZLhoHd7TicR4mOo8IIVriYCGAKW0A.w1JvTPO7IbZry.DxHrE3SN2BBzBJe0
Router(config-un)#commit
```

Running Configuration

```
username user1
policy test-policy1
secret 10
```

```
$6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZLhoHd7TicR4mOo8IIVriYCGAKW0A.w1JvTPO7IbZry.DxHrE3SN2BBzBJe0
!
```

The below examples show different possible combinations to check for password or secret compliance against the policy:

```
username user2
policy test-policy1
password 7 09604F0B
!
username user3
policy test-policy1
secret 10
$6$U3GZl1VINwJ4Dl1.$8X6av2kQ.AWvMKGEz5TLvZO7OXj6DgeOqLoQKIf7XJxKayViFJNateZ0no6gO6DbbXn4bBo/Dlqitro3jlsS40
password 7 080D4D4C
!
username user4
secret 10
$6$mA465X/m/UQ5....$rSKRw9B/SBYC/N.f7A9NCntPkrHXL6F4V26/NTjWXnrSnna03FxW3bcyfDAyveOexJz7/oak0XB6tjLF5CO981
password-policy test-policy1 password 7 0723204E
!
username user5
password-policy test-policy1 password 7 09604F0B
!
```

The compliance check for password or secret in the above examples works as described below:

- When you change the secret for user1, the system checks the secret compliance against the policy, test-policy1.
- When you change the password for user2, the system checks the password compliance against the policy, test-policy1.
- When you change the password or secret for user3, the system checks the password or secret compliance against the policy, test-policy1.
- When you change the secret for user4, the system does not check for compliance against any policy. Whereas, when you change the password for user4, the system checks the password compliance against the policy, test-policy1.
- When you change the password for user5, the system checks the password compliance against the policy, test-policy1.

The below example shows the order of configurations when performed in a single commit (say, by copy and paste). In such scenarios, if there is any username entry with a secret and policy configured, the system checks for secret compliance against that policy. In this example, the system does not check for any password compliance during the commit. So, the following configurations can be put in any order in a single commit.

```
(1)aaa password-policy pol1
 lifetime minutes 1
 upper-case 1
 restrict-old-count 2
!

username lab2
 group root-lr
(2) policy pol1
(3) secret 10
```

```
$6$gphqA0RfBXOn6A0.$wRwWG1l0TIpHPdVQ66fUiIM5P46ggoGMGgFuaZd0LD2DLFYDlDPaRyXQLi8Izjb49tC7H7tkTLrc1.GELFpiK.

  password 7 1533292F200F2D
!
```

Related Topics

- Password Policy for User Secret, on page 76

Associated Commands

- **aaa password-policy**
- **policy**
- **username**

Password Policy to Restrict Consecutive Characters

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Password Policy to Restrict Consecutive Characters	Release 7.7.1	We have enhanced the router security by enforcing a strong password policy for all users configured on the router. You can now specify a new password policy for the user that restricts the usage of a specific number of consecutive characters for the login passwords. These characters include English alphabets, the sequence of QWERTY keyboard layout, and numbers, such as, 'abcd', 'qwer', '1234', and so on. Apart from *passwords*, the feature is also applicable for *secrets*–the one-way encrypted secure login passwords that are not easy to decrypt to retrieve the original unencrypted password text. The password policy is applicable only for the users configured on the local AAA server on the router; not those configured on the remote AAA server. The feature introduces the **restrict-consecutive-characters** command.

Most often you create passwords and secrets which are easy to remember, such as the ones that use consecutive characters from English alphabets, or numbers. Such passwords and secrets are easy to compromise, thereby making the router vulnerable to security attacks. From Cisco IOS XR Software Release 7.7.1 and later, you can enhance the security of your user passwords and secrets by defining a password policy that restricts the usage of consecutive characters from English alphabets, QWERTY layout keyboard English alphabets, and numbers (such as, 'abcd', 'qwer', 'zyxw', '1234', and so on). You can also restrict a cyclic wrapping of the alphabet and the number (such as, 'yzab', 'opqw', '9012', and so on). The feature also gives you the flexibility to specify the number of consecutive alphabets or numbers to be restricted.

Certain key aspects of this feature are:

- The feature is disabled, by default.
- The security administrator must have *write* permission for AAA tasks to create the password policies.
- All password policies are applicable only to locally-configured users; not to users who are configured on remote AAA servers.

This table depicts the examples of valid and invalid passwords and secrets when the password policy to restrict consecutive characters (say, 4 in this example) is in place.

Use Case	Examples of Invalid Password and Secret	Examples of Valid Password and Secret
Restrict 4 consecutive English alphabets	AbcD, ABCD, TestPQRS, DcbA, TestZYxW123, DCBA, ihgf	AbcPqR, Xyzdef, Yzab, zabC
Restrict 4 consecutive English alphabets and decimal numbers from QWERTY keyboard layout	Qwer, QWER, Mnbv, aQwerm, Test1234, TestT7890, 5678, fghj	Opas, xzLk, sapo, saqw3210, Test9012
Restrict 4 consecutive English alphabets along with cyclic wrapping	Yzab, TestYZAB, zabc	1234, Qwer, QWER, Mnbv, aQwerm, Test1234, TestT0987
Restrict 4 consecutive English alphabets and numbers from QWERTY keyboard layout along with cyclic wrapping	9012, 8901, Test3210, TestT0987, Opqw, klas, dsal, Cxzm, nmzx	AbcD, ABCD, Yzab, TestYZAB, zabc

How to Restrict Consecutive Characters for User Passwords and Secrets

To enable the feature to restrict consecutive characters for user passwords and secrets, use the **restrict-consecutive-characters** command in *aaa password policy* configuration mode. To disable the feature, use the **no** form of the command.

You can use the optional keyword, **cyclic-wrap**, to restrict the cyclic wrapping of characters and numbers.

After creating the password policies, you must explicitly apply those policies to the user profiles so that the password policies take effect in the password and secret configuration.

Configuration Example

Enabling the feature using CLI:

```
Router(config)#aaa password-policy test-policy
Router(config-pp)#restrict-consecutive-characters english-alphabet 4
Router(config-pp)#restrict-consecutive-characters qwerty-keyboard 5
```

The keyword, **cyclic-wrap**, to restrict cyclic wrapping is an optional parameter. If configured, then the feature also restricts the cyclic wrapping of characters and numbers.

```
Router(config-pp)#restrict-consecutive-characters english-alphabet 4 cyclic-wrap
Router(config-pp)#restrict-consecutive-characters qwerty-keyboard 5 cyclic-wrap
```

Applying the password policy to the user profile:

```
Router(config)#username user1
Router(config-un)#policy test-policy
Router(config-un)#commit
```

Running Configuration

This is a sample running configuration that shows that you have configured a AAA password policy that restricts six consecutive characters from the QWERTY keyboard, and cyclic wrapping of four consecutive English alphabets.

```
Router(config-pp)#show running-config aaa password-policy
Tue May 17 10:53:16.532 UTC

!
aaa password-policy test-policy
 restrict-consecutive-characters qwerty-keyboard 6
 restrict-consecutive-characters english-alphabet 4 cyclic-wrap
!
```

Verification

You can use the **show aaa password-policy** command to know if the feature to restrict consecutive characters for user passwords and secrets is applied on the password policy.

```
Router#show aaa password-policy test-policy
Tue May 17 10:54:24.064 UTC
Password Policy Name : test-policy
  Number of Users : 0
  Minimum Length : 2
  Maximum Length : 253
  Special Character Len : 0
  Uppercase Character Len : 0
  Lowercase Character Len : 0
  Numeric Character Len : 0
  Policy Life Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 0
    years : 0
  Warning Interval :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
```

```
    months : 0
    years : 0
  Lockout Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 0
    years : 0
  Restrict Old Time :
    days : 0
    months : 0
    years : 0
  Character Change Len : 2
  Maximum Failure Attempts : 0
  Reference Count : 0
  Error Count : 0
  Lockout Count Attempts : 0
  Maximum char repetition : 0
  Restrict Old count : 0
  Restrict Username : 0
  Restrict Username Reverse  : 0
  Restrict Password Reverse  : 0
  Restrict Password Advanced : 0
  Restrict Consecutive Character :
    English Alphabet characters: 4
    English Alphabet Cyclic Wrap: True
    Qwerty Keyboard characters: 6
    Qwerty Keyboard Cyclic Wrap: False
Router#
```

Password or Secret Configuration Failure Scenarios:

You notice these logs or error messages on the router console when password or secret configuration fails because of the policy violation to restrict consecutive characters or numbers:

```
Router(config)#username user1
Router(config-un)#policy test-policy
Router(config-un)#password DEFg
Router(config-un)#commit
Tue Dec  7 10:17:56.843 UTC

% Failed to commit and rollback one or more configuration items. Please issue 'show
configuration failed [inheritance]' from this session to view the errors
Router(config-un)#show  configuration failed
username user1
 password 7 03205E0D01
!!% 'LOCALD' detected the 'fatal' condition 'Password contains consecutive characters from
 qwerty keyboard or English alphabet'
!
End


Router(config)#username user1
RP/0/RP0/CPU0:ios(config-un)#masked-secret
Fri Dec  3 12:33:44.354 UTC

Enter secret:
Re-enter secret:

secret is not compliant with policy to restrict consecutive letters or numbers
RP/0/RP0/CPU0:ios(config-un)#
```

```
Router(config)#username user1
Router(config-un)#policy test-policy
Router(config-un)#secret qwerty
                                    ^
% Invalid input detected at '^' marker.
Router(config-un)#
```

YANG Data Model to Restrict Consecutive Characters for User Passwords and Secrets

You can use the **Cisco-IOS-XR-aaa-locald-cfg** native YANG data model to restrict consecutive characters for user passwords and secrets. **Cisco-IOS-XR-um-aaa-locald-cfg** is the corresponding unified model (UM). You can access the data models from the Github repository.

The following is a sample format to enable the feature using the native YANG data model.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
 <target>
<candidate/>
</target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
 <aaa xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-lib-cfg">
<password-policies xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-cfg">
<password-policy>
     <name>test-policy</name>
     <restrict-consecutive-characters>
      <qwerty-keyboard>
       <characters>4</characters>
 <cyclic-wrap></cyclic-wrap>
      </qwerty-keyboard>
      <english-alphabet>
       <characters>4</characters>
       <cyclic-wrap></cyclic-wrap>
      </english-alphabet>
</restrict-consecutive-characters>
</password-policy>
</password-policies>
</aaa>
</config>
</edit-config>
</rpc>
##
```

To learn more about the data models and to put them to use, see the *Programmability Configuration Guide*.

Deprecation of Type 7 password and Type 5 secret

Password configuration options before Release 24.4.1

Until Release 24.4.1, there were two options for configuring a password:

- Password: Uses Type 7 encryption to store the password.
- Secret: Supports Type 5, 8, 9, or 10 hashing algorithms to store the password securely.

Deprecation Notice

Starting from the Release 24.4.1, the use of Type 7 password and Type 5 secret are deprecated due to security concerns. The deprecation process commences from the Release 24.4.1. We expect the full deprecation in a future release. We recommend using the default option, which is Type 10 secret.



Note With the deprecation of Type 7 password encryption in Cisco IOS XR Release 24.4.1, any configuration that used Type 7 passwords will be automatically converted and saved as Type 10 secrets during the upgrade to version 24.4.1.

If you have usernames that include both a password and a secret, then:

- For the first 3000 users, the router will retain the original secret and discard the password.
- For users beyond the first 3000, the router will convert the password as Type 10 secrets by overwriting the original secret.

password

The **password** options available in the router from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#password ?
LINE The type 7 password followed by '7 ' OR SHA512-based password (deprecated, use 'secret')
```

Changes:

- All the options that were present until the Release 24.4.1are removed except LINE (to accept cleartext).
- **During upgrade**: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.

  Post-upgrade: You can still use the Type 7 password configurations option after new commits, but the password will be stored as Type 10 secret.
- New **syslog** has been added to indicate the deprecation process:

  ```
  %SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is deprecated.

   Converting it to a Type 10 secret for user <user name>.
  ```
- **show running configuration** command output before upgrade:

  ```
  username example
  password 7 106D000A0618
  !
  ```

  show running configuration command output post-upgrade:

  ```
  username example
  Cisco Confidential
  secret 10
  $6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iClOryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqq8st.
  !
  ```

masked-password

The **masked-password** options available in CLI from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#masked-password ?
0 Specifies a cleartext password will follow
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
<cr> The cleartext user password
```

Changes:

- The options 7 and encrypted that were present until the Release 24.4.1 are removed.
- **During upgrade**: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.
- **Post-upgrade**: Masked-password is an alternate method of configuring the password. You can still use the masked-password keyword with a clear string after new commits, but the password will be stored as Type 10 secret.
- New **syslog** has been added to indicate the deprecation process:

  ```
  %SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is deprecated.

   Converting it to a Type 10 secret for user <user name>.
  ```

- **show running configuration** command output before upgrade:

  ```
  username example
  password 7 106D000A0618
  !
  ```

  show running configuration command output post-upgrade:

  ```
  username example
  Cisco Confidential
  secret 10
  $6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iClOryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqq8st.
  !
  ```

password-policy

The **password-policy** options available in CLI from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#password-policy ?
WORD Specify the password policy name

RP/0/RP0/CPU0:ios(config-un)#password-policy abcd password ?
0 Specifies an UNENCRYPTED password will follow
7 Specifies that an encrypted password will follow
LINE The UNENCRYPTED (cleartext) user password
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
encrypted Config deprecated. Will be removed in 7.7.1. Specify '7' instead.
```

Changes:

- All the options that were present until 24.4.1are removed except LINE (to accept cleartext).
- **During upgrade**: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.

  Post-upgrade: You can still use the password-policy configurations option after new commits, but the it will be stored as Type 10 secret.

- New **syslog** has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is deprecated.

 Converting it to a Type 10 secret for user <username>.
```

- **show running configuration**f command output before upgrade:

```
username example
password-policy abcd password 7 106D000A0618
!
```

show running configuration command output post-upgrade:

```
username example
secret 10
$6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iClOryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqq8st.
!
!
```

aaa password-policy

The **aaa password-policy** options available in CLI from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config)#aaa password-policy abcd
RP/0/RP0/CPU0:ios(config-pp)#?
min-char-change Number of characters change required between old and new passwords
(deprecated, will be removed in 25.3.1)
restrict-password-advanced Advanced restrictions on new password (deprecated, will be removed
 in 25.3.1)
restrict-password-reverse Restricts the password to be same as reversed old password
(deprecated, will be removed in 25.3.1)
```

Changes:

- The options min-char-change, restrict-password-advanced, and restrict-password-reverse that were present until the Release 24.4.1 are deprecated.
- **During upgrade**: These deprecated configurations do not go through any change during upgrade.

  Post-upgrade: These deprecated keywords do not take effect when configured post-upgrade.
- New **syslog** have been added to indicate the deprecation process:

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy option
'min-char-change' is deprecated.
Password/Secret will not be checked against this option now.
```

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy option
'restrict-password-reverse' is deprecated.
 Password/Secret will not be checked against this option now.
```

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy option
'restrict-password-advanced' is deprecated.
Password/Secret will not be checked against this option now.
```

- **show running configuration** command output before upgrade:

```
aaa password-policy abcd
lower-case 3
min-char-change 1
restrict-password-reverse
restrict-password-advanced
!
```

show running configuration command output post-upgrade:

```
aaa password-policy abcd
lower-case 3
min-char-change 1
restrict-password-reverse
restrict-password-advanced
!
```

secret

The **secret** options available in CLI from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#secret ?
0 Specifies a cleartext password will follow
10 Specifies that SHA512-based password will follow
8 Specifies that SHA256-based password will follow
9 Specifies that Scrypt-based password will follow
LINE The cleartext user password
```

```
RP/0/RP0/CPU0:ios(config-un)#secret 0 enc-type ?
<8-10> Specifies which algorithm to use. Only 8,9,10 supported [Note: Option ‘5’ is not
available to use from 24.4]
```

Changes:

- The options 5 and encrypted are removed.
- **During upgrade**: Configurations using Type 5 secret will remain unchanged.

  Post-upgrade: Though the keyword 5 has been deprecated, you can still apply the existing configurations using Type 5 secret.
- New **syslog** has been added to indicate the deprecation process:

  ```
  %SECURITY-LOCALD-2-DEPRECATED_SECRET_TYPE : Type 5 secret is deprecated.
  Please use the 'secret' keyword with option type 10 for user.
  ```
- **show running configuration** command output before upgrade:

  ```
  username example
  secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
  !
  !
  ```

  show running configuration command output post-upgrade:

  ```
  username example
  secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
  !
  !
  ```

masked-secret

The **masked-secret** options available in CLI from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#masked-secret ?
0 Specifies a cleartext password will follow
Cisco Confidential
10 Specifies that SHA512-based password will follow
8 Specifies that SHA256-based password will follow
9 Specifies that Scrypt-based password will follow
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
<cr> The cleartext user password
```

Changes:

- The options 5 and encrypted are removed.
- **During upgrade**: Configurations using masked-secret with Type 5 will remain unchanged.

  Post-upgrade: Though the keyword 5 has been deprecated, you can still apply the existing configurations using Type 5 masked secret.
- New **syslog** has been added to indicate the deprecation process:

```
%SECURITY-LOCALD-2-DEPRECATED_SECRET_TYPE : Type 5 secret is deprecated.
Please use the 'secret' keyword with option type 10 for user.
```

- **show running configuration** command output before upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
!
!
```

  show running configuration command output post-upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
!
!
```

Special use cases

Use case 1: Configurations using both Type 7 password and secret with 8, 9, or 10 hashing, for the same user

- **During upgrade**:
  - For the first 3000 username configurations, the password configuration will be rejected, and the secret configuration will remain unchanged.
  - For the rest of the username configurations, the original secret configuration will be rejected, and the password will be converted to Type 10 secret.
- **Post-upgrade**:
  - For a new username configured, or the username that is already present before the upgrade, the password configuration will be rejected.
  - New **syslog** has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-SECRET_CONFIG_PRESENT : The password configuration is deprecated.

Once secret is configured, cannot use password config for user <user name> at index
 <x> now.
```

    where 'x' is a number representing the index.

Use case 2: Configurations using both Type 7 password and Type 5 secret, for the same user

- **During upgrade**:
  - For any username configuration, the original Type 5 secret configuration will be rejected, and the password will be converted to Type 10 secret.
- **Post-upgrade**:

- For a new username configured, or the username that is already present before the upgrade, the password configuration will be converted to Type 10 secret.
- New **syslog** has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is
deprecated.
 Converting it to a Type 10 secret for user <username>.
```

Configure Router to RADIUS Server Communication

This task configures router to RADIUS server communication. The RADIUS host is normally a multiuser system running RADIUS server software from Cisco (CiscoSecure ACS), Livingston, Merit, Microsoft, or another software provider. Configuring router to RADIUS server communication can have several components:

- Hostname or IP address
- Authentication destination port
- Accounting destination port
- Retransmission value
- Timeout period
- Key string

RADIUS security servers are identified on the basis of their hostname or IP address, hostname and specific User Datagram Protocol (UDP) port numbers, or IP address and specific UDP port numbers. The combination of the IP address and UDP port numbers creates a unique identifier, allowing different ports to be individually defined as RADIUS hosts providing a specific AAA service. In other words, this unique identifier enables RADIUS requests to be sent to multiple UDP ports on a server at the same IP address. If two different host entries on the same RADIUS server are configured for the same service—for example, accounting—the second host entry configured acts as an automatic switchover backup to the first one. Using this example, if the first host entry fails to provide accounting services, the network access server tries the second host entry configured on the same device for accounting services. (The RADIUS host entries are tried in the order they are configured.)

A RADIUS server and a Cisco router use a shared secret text string to encrypt passwords and exchange responses.To configure RADIUS to use the AAA security commands, you must specify the host running the RADIUS server daemon and a secret text (key) string that it shares with the router.

The timeout, retransmission, and encryption key values are configurable globally for all RADIUS servers, on a per-server basis, or in some combination of global and per-server settings. To apply these settings globally to all RADIUS servers communicating with the router, use the three unique global commands: **radius-server timeout**, **radius-server retransmit**, and **radius-server key**. To apply these values on a specific RADIUS server, use the **radius-server host** command.

You can configure a maximum of 30 global RADIUS servers.



Note You can configure both global and per-server timeout, retransmission, and key value commands simultaneously on the same Cisco network access server. If both global and per-server functions are configured on a router, the per-server timer, retransmission, and key value commands override global timer, retransmission, and key value commands.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **radius-server host** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*]

Example:

```
RP/0//CPU0:router(config)# radius-server host host1
```

Specifies the hostname or IP address of the remote RADIUS server host.

- Use the **auth-port** *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for authentication.
- Use the **acct-port** *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for accounting.
- To configure the network access server to recognize more than one host entry associated with a single IP address, simply repeat this command as many times as necessary, making sure that each UDP port number is different. Set the timeout, retransmit, and encryption key values to use with the specific RADIUS host.
- If no timeout is set, the global value is used; otherwise, enter a value in the range 1 to 1000. If no retransmit value is set, the global value is used; otherwise enter a value in the range 1 to 100. If no key string is specified, the global value is used.

Note
The key is a text string that must match the encryption key used on the RADIUS server. Always configure the key as the last item in the **radius-server host** command syntax because the leading spaces are ignored, but spaces within and at the end of the key are used. If you use spaces in your key, do not enclose the key in quotation marks unless the quotation marks themselves are part of the key.

Step 3 **radius-server retransmit** *retries*

Example:

```
RP/0/RP0/CPU0:router(config)# radius-server retransmit 5
```

Specifies the number of times the software searches the list of RADIUS server hosts before giving up.

- In the example, the number of retransmission attempts is set to 5.

Step 4 **radius-server timeout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config)# radius-server timeout 10
```

Sets the number of seconds a router waits for a server host to reply before timing out.

- In the example, the interval timer is set to 10 seconds.

Step 5 **radius-server key** {**0** *clear-text-key* | **7** *encrypted-key* | *clear-text-key*}

Example:

```
RP/0/RP0/CPU0:router(config)# radius-server key 0 samplekey
```

Sets the authentication and encryption key for all RADIUS communications between the router and the RADIUS daemon.

Step 6 **radius source-interface** *type instance* [**vrf** *vrf-id*]

Example:

```
RP/0/RP0/CPU0:router(config)# radius source-interface 0/3/0/1
```

(Optional) Forces RADIUS to use the IP address of a specified interface or subinterface for all outgoing RADIUS packets.

- The specified interface or subinterface must have an IP address associated with it. If the specified interface or subinterface does not have an IP address or is in the down state, then RADIUS reverts to the default. To avoid this, add an IP address to the interface or subinterface or bring the interface to the up state.

The **vrf** keyword enables the specification on a per-VRF basis.

Step 7 Repeat step 2 through step 6 for each external server to be configured.

—

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 9 show radius

Example:

```
RP/0/RP0/CPU0:router# show radius
```

(Optional) Displays information about the RADIUS servers that are configured in the system.

Radius Summary Example

```
radius source-interface Mgm0/rp0/cpu0/0 vrf default
radius-server timeout 10
radius-server retransmit 2
!
! OOB RADIUS
radius-server host 123.100.100.186 auth-port 1812 acct-port 1813
```

```
key cisco123
timeout 10
retransmit 2
!
radius-server host 123.100.100.187 auth-port 1812 acct-port 1813
key cisco123
timeout 10
retransmit 2
!
aaa group server radius radgrp
server 123.100.100.186 auth-port 1812 acct-port 1813
server 123.100.100.187 auth-port 1812 acct-port 1813
!
aaa authorization exec radauthen group radgrp local
aaa authentication login radlogin group radgrp local
!
line template vty
authorization exec radauthen
login authentication radlogin
timestamp disable
exec-timeout 0 0
!
vty-pool default 0 99 line-template vty
```

Configure RADIUS Dead-Server Detection

The RADIUS Dead-Server Detection feature lets you configure and determine the criteria that is used to mark a RADIUS server as dead. If no criteria is explicitly configured, the criteria is computed dynamically on the basis of the number of outstanding transactions. The RADIUS dead-server detection configuration results in the prompt detection of RADIUS servers that have stopped responding. The prompt detection of nonresponding RADIUS servers and the avoidance of swamped and dead-to-live-to-dead-again servers result in less deadtime and quicker packet processing.

You can configure the minimum amount of time, in seconds, that must elapse from the time that the router last received a valid packet from the RADIUS server to the time the server is marked as dead. If a packet has not been received since the router booted, and there is a timeout, the time criterion is treated as though it was met.

In addition, you can configure the number of consecutive timeouts that must occur on the router before the RADIUS server is marked as dead. If the server performs both authentication and accounting, both types of packets are included in the number. Improperly constructed packets are counted as though they are timeouts. Only retransmissions are counted, not the initial transmission. For example, each timeout causes one retransmission to be sent.

Note Both the time criterion and the tries criterion must be met for the server to be marked as dead.

The **radius-server deadtime** command specifies the time, in minutes, for which a server is marked as dead, remains dead, and, after this period, is marked alive even when no responses were received from it. When the dead criteria are configured, the servers are not monitored unless the **radius-server deadtime** command is configured

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **radius-server deadtime** *minutes*

Example:

```
RP/0/RP0/CPU0:router(config)# radius-server deadtime 5
```

Improves RADIUS response times when some servers might be unavailable and causes the unavailable servers to be skipped immediately.

Step 3 **radius-server dead-criteria time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config)# radius-server dead-criteria time 5
```

Establishes the time for the dead-criteria conditions for a RADIUS server to be marked as dead.

Step 4 **radius-server dead-criteria tries** *tries*

Example:

```
RP/0/RP0/CPU0:router(config)# radius-server dead-criteria tries 4
```

Establishes the number of tries for the dead-criteria conditions for a RADIUS server to be marked as dead.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 6 **show radius dead-criteria host** *ip-addr* [**auth-port** *auth-port*] [**acct-port** *acct-port*]

Example:

```
RP/0/RP0/CPU0:router# show radius dead-criteria host 172.19.192.80
```

(Optional) Displays dead-server-detection information that has been requested for a RADIUS server at the specified IP address.

Configure TACACS+ Server

This task configures a TACACS+ server.

The port, if not specified, defaults to the standard port number, 49. The **timeout** and **key** parameters can be specified globally for all TACACS+ servers. The **timeout** parameter specifies how long the AAA server waits to receive a response from the TACACS+ server. The **key** parameter specifies an authentication and encryption key shared between the AAA server and the TACACS+ server.

The **single-connection** parameter specifies to multiplex all TACACS+ requests to the TACACS+ server over a single TCP connection. The **single-connection-idle-timeout** parameter specifies the timeout value for this single connection.

You can configure a maximum of 30 global TACACS+ servers.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **tacacs-server host** *host-name* **port** *port-number*

Example:

```
RP/0/RP0/CPU0:router(config)# tacacs-server host 209.165.200.226 port 51
RP/0/RP0/CPU0:router(config-tacacs-host)#
```

Specifies a TACACS+ host server and optionally specifies a server port number.

- This option overrides the default, port 49. Valid port numbers range from 1 to 65535.

Step 3 **tacacs-server host** *host-name* **timeout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-tacacs-host)# tacacs-server host 209.165.200.226 timeout 30
RP/0/RP0/CPU0:router(config)#
```

Specifies a TACACS+ host server and optionally specifies a timeout value that sets the length of time the AAA server waits to receive a response from the TACACS+ server.

- This option overrides the global timeout value set with the **tacacs-server timeout** command for only this server. The timeout value is expressed as an integer in terms of timeout interval seconds. The range is from 1 to 1000.

Step 4 **tacacs-server host** *host-name* **key** [**0** | **7**] *auth-key*

Example:

```
RP/0/RP0/CPU0:router(config)# tacacs-server host 209.165.200.226 key 0 a_secret
```

Specifies a TACACS+ host server and optionally specifies an authentication and encryption key shared between the AAA server and the TACACS+ server.

- The TACACS+ packets are encrypted using this key. This key must match the key used by TACACS+ daemon. Specifying this key overrides the global key set by the **tacacs-server key** command for only this server.
- (Optional) Entering **0** indicates that an unencrypted (clear-text) key follows.
- (Optional) Entering **7** indicates that an encrypted key follows.
- The *auth-key* argument specifies the encrypted or unencrypted key to be shared between the AAA server and the TACACS+ server.

Step 5 **tacacs-server host** *host-name* **single-connection**

Example:

```
RP/0/RP0/CPU0:router(config)# tacacs-server host 209.165.200.226 single-connection
```

Prompts the router to multiplex all TACACS+ requests to this server over a single TCP connection. By default, a separate connection is used for each session.

Step 6 **tacacs-server host** *host-name* **single-connection-idle-timeout** *timeout-in-seconds*

Example:

```
RP/0/0RP0RSP0/CPU0:router:hostname(config)#tacacs-server host 209.165.200.226
single-connection-idle-timeout 60
```

Sets the timeout value, in seconds, for the single TCP connection (that is created by configuring the **single-connection** command) to the TACACS+ server.

The range is:

- 500 to 7200 (prior to Cisco IOS XR Software Release 7.4.1/Release 7.3.2)
- 5 to 7200 (from Cisco IOS XR Software Release 7.4.1/Release 7.3.2, and later)

Step 7 **tacacs source-interface** *type instance*

Example:

```
RP/0/RP0/CPU0:router(config)# tacacs source-interface 0/4/0/0
```

(Optional) Specifies the source IP address of a selected interface for all outgoing TACACS+ packets.

- The specified interface or subinterface must have an IP address associated with it. If the specified interface or subinterface does not have an IP address or is in the down state, then TACACS+ reverts to the default interface. To avoid this, add an IP address to the interface or subinterface or bring the interface to the up state.
- The **vrf** option specifies the Virtual Private Network (VPN) routing and forwarding (VRF) reference of an AAA TACACS+ server group.

Step 8 Repeat step 2 through step 6 for each external server to be configured.

—

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show tacacs**

Example:

```
RP/0/RP0/CPU0:router# show tacacs
```

(Optional) Displays information about the TACACS+ servers that are configured in the system.

Tacacs Summary Example:

```
! OOB TAC
tacacs-server host 123.100.100.186 port 49
key lm51
!
tacacs-server host 123.100.100.187 port 49
key lm51
!
aaa group server tacacs+ tacgrp
server 123.100.100.186
server 123.100.100.187
!
aaa group server tacacs+ eem
server 123.100.100.186
server 123.100.100.187
!
aaa authorization exec tacauthen group tacgrp local
aaa authentication login taclogin group tacgrp local
!
line console
authorization exec tacauthen
login authentication taclogin
timeout login response 30
timestamp
exec-timeout 0 0
session-timeout 15
!
vty-pool default 0 99 line-template console
```

Configure RADIUS Server Groups

This task configures RADIUS server groups.

The user can enter one or more **server** commands. The **server** command specifies the hostname or IP address of an external RADIUS server along with port numbers. When configured, this server group can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

You can configure a maximum of:

- 30 servers per RADIUS server group

- 30 private servers per RADIUS server group

Before you begin

For configuration to succeed, the external server should be accessible at the time of configuration.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **aaa group server radius** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config)# aaa group server radius radgroup1
```

Groups different server hosts into distinct lists and enters the server group configuration mode.

Step 3 **server** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*]

Example:

```
RP/0/RP0/CPU0:router(config-sg-radius)# server 192.168.20.0
```

Specifies the hostname or IP address of an external RADIUS server.

- After the server group is configured, it can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

Step 4 Repeat step 4 for every external server to be added to the server group named in step 3.

—

Step 5 **deadtime** *minutes*

Example:

```
RP/0/RP0/CPU0:router(config-sg-radius)# deadtime 1
```

Configures the deadtime value at the RADIUS server group level.

- The *minutes* argument specifies the length of time, in minutes, for which a RADIUS server is skipped over by transaction requests, up to a maximum of 1440 (24 hours). The range is from 1 to 1440.

The example specifies a one-minute deadtime for RADIUS server group radgroup1 when it has failed to respond to authentication requests for the **deadtime** command

Note
You can configure the group-level deadtime after the group is created.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 7 **show radius server-groups** [*group-name* [**detail**]]

Example:

```
RP/0/RP0/CPU0:router# show radius server-groups
```

(Optional) Displays information about each RADIUS server group that is configured in the system.

What to do next

After configuring RADIUS server groups, define method lists by configuring authentication, authorization, and accounting.

Configure TACACS+ Server Groups

This task configures TACACS+ server groups.

You can enter one or more **server** commands. The **server** command specifies the hostname or IP address of an external TACACS+ server. Once configured, this server group can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

Before you begin

For successful configuration, the external server should be accessible at the time of configuration. When configuring the same IP address for global and vrf configuration, server-private parameters are required (see *Configure Per VRF TACACS+ Server Groups* section).

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **aaa group server tacacs+** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config)# aaa group server tacacs+ tacgroup1
```

Groups different server hosts into distinct lists and enters the server group configuration mode.

Step 3 **server** {*hostname* | *ip-address*}

Example:

```
RP/0/RP0/CPU0:router(config-sg-tacacs+)# server 192.168.100.0
```

Specifies the hostname or IP address of an external TACACS+ server.

- When configured, this group can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

Step 4 Repeat step 3 for every external server to be added to the server group named in step 2.

—

Step 5 **server-private** {*hostname* | *ip-address in IPv4 or IPv6 format*} [**port** *port-number*] [**timeout** *seconds*] [**key** *string*]

Example:

```
Router(config-sg-tacacs+)# server-private 10.1.1.1 key a_secret
```

Configures the IP address of the private TACACS+ server for the group server.

Note

- You can configure a maximum of 10 TACACS+ servers per server group.
- You can configure a maximum of 10 private TACACS+ servers.
- If private server parameters are not specified, global configurations are used. If global configurations are not specified, default values are used.

Step 6 (Optional) **vrf** *vrf-id*

Example:

```
Router(config-sg-tacacs+)# vrf test-vrf
```

The vrf option specifies the Virtual Private Network (VPN) routing and forwarding (VRF) reference of an AAA TACACS+ server group.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8 **show tacacs server-groups**

Example:

```
RP/0/RP0/CPU0:router# show tacacs server-groups
```

(Optional) Displays information about each TACACS+ server group that is configured in the system.

Configure Per VRF TACACS+ Server Groups

The Cisco IOS XR software supports per VRF AAA to be configured on TACACS+ server groups. You must use the **server-private** and **vrf** commands as listed below to configure this feature.

The global server definitions can be referred from multiple server groups, but all references use the same server instance and connect to the same server. In case of VRF, you do not need the global configuration because the server status, server statistics and the key could be different for different VRFs. Therefore, you must use the server-private configuration if you want to configure per VRF TACACS+ server groups. If you have the same server used in different groups with different VRFs, ensure that it is reachable through all those VRFs.

If you are migrating the servers to a VRF, then it is safe to remove the global server configuration with respect to that server.

Prerequisites

You must ensure these before configuring per VRF on TACACS+ server groups:

- Be familiar with configuring TACACS+, AAA, per VRF AAA, and group servers.
- Ensure that you have access to the TACACS+ server.
- Configure the VRF instance before configuring the specific VRF for a TACACS+ server and ensure that the VRF is reachable.

Configuration Example

```
Router#configure

/* Groups different server hosts into distinct lists and enters the server group configuration
 mode.
You can enter one or more server commands. The server command specifies the hostname or IP
 address of an external TACACS+ server.
Once configured, this server group can be referenced from the AAA method lists (used while
 configuring authentication, authorization, or accounting). */

Router(config)# aaa group server tacacs+ tacgroup1

/* Configures the IP address and the secret key of the private TACACS+ server that is
reachable through specific VRF.
You can have multiple such server configurations which are reachable through the same VRF.*/

Router(config-sg-tacacs+)# server-private 10.1.1.1 port 49 key a_secret

/* The vrf option specifies the VRF reference of a AAA TACACS+ server group */
Router(config-sg-tacacs+)# vrf test-vrf
Router(config-sg-tacacs+)# commit
```

Running Configuration

```
aaa group server tacacs+ tacgroup1
 vrf test-vrf
 server-private 10.1.1.1 port 49
  key 7 0822455D0A16
 !
 server-private 10.1.1.2 port 49
  key 7 05080F1C2243
```

```
 !
 server-private 2001:db8:1::1 port 49
  key 7 045802150C2E
 !
 server-private 2001:db8:1::2 port 49
  key 7 13061E010803
 !
!
```

Verify Per VRF TACACS+ Server Groups

```
Router#show tacacs
Fri Sep 27 11:14:34.991 UTC

Server: 10.1.1.1/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4

Server: 10.1.1.2/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4

Server: 2001:db8:1::1/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv6

Server: 2001:db8:1::2/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv6
```

Associated Commands

- **server-private**
- **vrf**

TACACS+ with TLS protection

The TACACS+ with TLS protection is a security enhancement to the TACACS+ protocol that

- uses Transport Layer Security (TLS) to encrypt authentication, authorization, and accounting (AAA) communication between network devices and TACACS+ servers,
- provides confidentiality and integrity for sensitive data transmitted over potentially insecure networks, and
- supports mutual authentication to safeguard against unauthorized access.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
TACACS+ with TLS protection	Release 25.3.1	You can significantly enhance security and reduce the risk of attacks on weak encryption by using TACACS+ over TLS. This method ensures the secure transmission of all Authentication, Authorization, and Accounting (AAA) data between the client and server. It provides robust protection for sensitive environments by supporting mutual authentication through a TLS X.509 certificate-based infrastructure. This feature is compatible with both TLS versions 1.3 and 1.2.

Benefits of TACACS+ with TLS protection

- Enhances security by encrypting TACACS+ communications.
- Protects AAA data with robust encryption.
- Supports mutual authentication between client and server.
- Complies with TLS 1.3 and 1.2 standards.

TACACS+ with TLS protection improves the security of AAA services by using TLS for encrypted communication. It addresses vulnerabilities associated with weak encryption and is designed for sensitive environments that require strong protection for AAA data.

How TACACS+ with TLS protection work

Summary

TACACS+ with TLS protection secures AAA communications between a network device and a TACACS+ server by carrying TACACS+ packets within an encrypted TLS session that is established when a user initiates SSH access.

The key components involved in the process are:

- End user: Initiates an SSH session to access the network device.
- Router as TACACS+ TLS client: The router receives the AAA service request and initiates a TLS session to the TACACS+ server.
- TACACS+ server: The server uses valid TLS configuration, terminates the TLS session, and processes AAA requests and responses.
- TLS session: Encrypts the TACACS+ traffic between the client and server.
- TACACS+ packets: Carry authentication, authorization, and accounting information over the TLS tunnel.

Workflow

These are the stages of how TACACS+ with TLS protection works:

1. Session initiation: The end user initiates an SSH session to the network device.

2. AAA request: The router receives the user's AAA service request for TACACS+ with TLS protection.
3. TLS establishment: If the TACACS+ server is configured with valid TLS settings, the TACACS+ TLS client and server establish a TLS session.
4. Secure exchange: The client and server exchange TACACS+ packets over the TLS-encrypted session.

Result

The process establishes an encrypted channel for TACACS+ traffic, securing AAA communications during SSH access.

Guidelines for TACACS+ with TLS protection

Follow these guidelines when configuring TACACS+ with TLS protection:

- Configure the destination port for TACACS+ with TLS protection. There are no dedicated ports for authentication, accounting, or authorization changes.
- Use TLS X.509 certificate-based mutual authentication between client and server.
- Ensure you use either TLS 1.3 (as mandated by RFC 8446) or TLS 1.2, which is also supported.
- Use either multi-connect or single connect without TLS resumption as needed.
- Ensure your implementation supports the cipher suites mandated by TLS 1.3 and TLS 1.2.
- Use IPv4 or IPv6 for TACACS+ transactions as appropriate.
- Use the source interface and non-default Virtual Routing and Forwarding (VRF) as required.
- Configure the connection timer and single-connect idle timeout as appropriate—these settings work the same as for TACACS+ over TCP.

Restrictions for TACACS+ with TLS protection

These restrictions apply when configuring TACACS+ with TLS protection:

- The TACACS+ encryption method supported in Cisco IOS-XR software releases before 25.3.1 is no longer supported.
- The router does not support using both non-TLS and TLS servers in the same server group.
- The router does not support TLS session resumption or TLS sessions with PSK cipher suites.

Configure TACACS+ with TLS protection

Configure TACACS+ to use Transport Layer Security (TLS) for secure communication, enhancing the confidentiality and integrity of authentication, authorization, and accounting traffic.

This task describes how to enable TLS for TACACS+ server communication. You can enable TLS directly for a TACACS+ host or within an AAA server group. With TLS enabled, data exchanged between the network device and the TACACS+ server is encrypted.

Procedure

Step 1 Use the **tls** to enable TACACS+ with TLS protection.

a) Tacacs-server host configuration:

Example:

```
Router(config)# tacacs-server host 10.105.236.101 port 4950
Router(config-tacacs-host)# tls
Router(config-tacacs-host-tls)# server-name-indicator aaa.cisco.com
Router(config-tacacs-host-tls)# trustpoint test
Router(config-tacacs-host-tls)# commit
```

b) Server-group configuration:

Example:

```
Router(config)# configure
Router(config)# aaa group server tacacs+ tac1
Router(config-sg-tacacs)# server-private 10.105.236.101 port 2345
Router(config-sg-tacacs-private)# tls
Router(config-sg-tacacs-private-tls)# server-name-indicator aaa.cisco.com
Router(config-sg-tacacs-private-tls)# trustpoint abc
Router(config-sg-tacacs-private-tls)# commit
```

Step 2 Run the **show tacacs** command to display the TACACS information.

Example:

```
Router# show tacacs
Info: Verify that TACACS+ with TLS protection is configured.

Server: 10.105.236.101/2084
.....
FIPS mode : TRUE/FALSE.
TLS:
   Version: TLS 1.3/1.2      // only for active connection
   Cipher: TLS_AES_128_GCM_SHA256  // only for active connection
Statistics:
   Successfull connections: 0
   Failed connections         : 0
   SSL  errors :
    Connect error:
    Read error:
    Write error:
    Handshake Failure:
    Protocol Mismatch :
  Certificate Validation Error:
  Cipher Suite Mismatch:
  Session Timeout:
  Revoked Certificate:
  Unsupported TLS Version:
  Untrusted CA:
```

TACACS+ communication with the specified server is now secured using TLS, encrypting authentication, authorization, and accounting traffic.

Create Series of Authentication Methods

Authentication is the process by which a user (or a principal) is verified. Authentication configuration uses *method lists* to define an order of preference for the source of AAA data, which may be stored in a variety of data sources. You can configure authentication to define more than one method list and applications (such as login) can choose one of them. For example, console ports may use one method list and the vty ports may use another. If a method list is not specified, the application tries to use a default method list.



Note Applications should explicitly refer to defined method lists for the method lists to be effective.

The authentication can be applied to tty lines through use of the **login authentication** line configuration submode command. If the method is RADIUS or TACACS+ servers, rather than server group, the RADIUS or TACACS+ server is chosen from the global pool of configured RADIUS and TACACS+ servers, in the order of configuration. Servers from this global pool are the servers that can be selectively added to a server group.

The subsequent methods of authentication are used only if the initial method returns an error, not if the request is rejected.

Before you begin



Note The default method list is applied for all the interfaces for authentication, except when a non-default named method list is explicitly configured, in which case the named method list is applied.

The **group radius, group tacacs+**, and **group** *group-name* forms of the **aaa authentication** command refer to a set of previously defined RADIUS or TACACS+ servers. Use the **radius server-host or tacacs-server host** command to configure the host servers. Use the **aaa group server radius or aaa group server tacacs+** command to create a named group of servers.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **aaa authentication** {**login**} {**default** | *list-name*} *method-list*

Example:

```
RP/0//CPU0:router(config)# aaa authentication login default group tacacs+
```

Creates a series of authentication methods, or a method list.

- Using the **login** keyword sets authentication for login. Using the **ppp** keyword sets authentication for Point-to-Point Protocol.
- Entering the **default** keyword causes the listed authentication methods that follow this keyword to be the default list of methods for authentication.
- Entering a *list-name* character string identifies the authentication method list.
- Entering a *method-list* argument following the method list type. Method list types are entered in the preferred sequence. The listed method types are any one of the following options:
  - **group tacacs+**—Use a server group or TACACS+ servers for authentication
  - **group radius**—Use a server group or RADIUS servers for authentication
  - **group** *named-group*—Use a named subset of TACACS+ or RADIUS servers for authentication
  - **local**—Use a local username or password database for authentication
  - **line**—Use line password or user group for authentication
- The example specifies the **default** method list to be used for authentication.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 4 Repeat Step 1 through Step 3 for every authentication method list to be configured.

—

Create Series of Authorization Methods

Method lists for authorization define the ways authorization will be performed and the sequence in which these methods will be performed. A method list is a named list describing the authorization methods to be used (such as TACACS+), in sequence. Method lists enable you to designate one or more security protocols to be used for authorization, thus ensuring a backup system if the initial method fails. The software uses the first method listed to authorize users for specific network services; if that method fails to respond, the software selects the next method listed in the method list. This process continues until there is successful communication with a listed authorization method, or until all methods defined have been exhausted.



Note The software attempts authorization with the next listed method only when there is no response or an error response (not a failure) from the previous method. If authorization fails at any point in this cycle—meaning that the security server or local username database responds by denying the user services—the authorization process stops and no other authorization methods are attempted.

When you create a named method list, you are defining a particular list of authorization methods for the indicated authorization type. When defined, method lists must be applied to specific lines or interfaces before any of the defined methods are performed. Do not use the names of methods, such as TACACS+, when creating a new method list.

"Command" authorization, as a result of adding a command authorization method list to a line template, is separate from, and is in addition to, "task-based" authorization, which is performed automatically on the router. The default behavior for command authorization is none. Even if a default method list is configured, that method list has to be added to a line template for it to be used.

The **aaa authorization commands** command causes a request packet containing a series of attribute value (AV) pairs to be sent to the TACACS+ daemon as part of the authorization process. The daemon can do one of the following:

- Accept the request as is.
- Refuse authorization.



Note To avoid lockouts in user authorization, make sure to allow local fallback (by configuring the **local** option for **aaa authorization** command) when configuring AAA. For example, **aaa authorization commands default tacacs+ local**.

Use the **aaa authorization** command to set parameters for authorization and to create named method lists defining specific authorization methods that can be used for each line or interface.



Note If you have configured AAA authorization to be subjected to TACACS+ authorization, then you must ensure that the server group is configured (use the **aaa group server tacacs+** command for this) for that TACACS+ server. Else, authorization fails.

For example,

```
aaa authorization exec default group test_tacacs+ local
aaa authorization commands default group test_tacacs+
aaa group server tacacs+ test_tacacs+ <===
```

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **aaa authorization** {**commands** | **eventmanager** | **exec** | **network**} {**default** | *list-name*} {**none** | **local** | **group** {**tacacs+** | **radius** | *group-name*}}

Example:

```
RP/0//CPU0:router(config)# aaa authorization commands listname1 group tacacs+
```

Creates a series of authorization methods, or a method list.

- The **commands** keyword configures authorization for all XR EXEC mode shell commands. Command authorization applies to the EXEC mode commands issued by a user. Command authorization attempts authorization for all XR EXEC modecommands.
- The **eventmanager** keyword applies an authorization method for authorizing an event manager (fault manager).
- The **exec** keyword configures authorization for an interactive (XR EXEC mode) session.
- The **network** keyword configures authorization for network services like PPP or IKE.
- The **default** keyword causes the listed authorization methods that follow this keyword to be the default list of methods for authorization.
- A *list-name* character string identifies the authorization method list. The method list itself follows the method list name. Method list types are entered in the preferred sequence. The listed method list types can be any one of the following:
  - **none**—The network access server (NAS) does not request authorization information. Authorization always succeeds. No subsequent authorization methods will be attempted. However, the task ID authorization is always required and cannot be disabled.
  - **local**—Uses local database for authorization.
- **group tacacs+**—Uses the list of all configured TACACS+ servers for authorization. The NAS exchanges authorization information with the TACACS+ security daemon. TACACS+ authorization defines specific rights for users by associating AV pairs, which are stored in a database on the TACACS+ security server, with the appropriate user.
- **group radius**—Uses the list of all configured RADIUS servers for authorization.
- **group** *group-name*—Uses a named server group, a subset of TACACS+ or RADIUS servers for authorization as defined by the **aaa group server tacacs+** or **aaa group server radius** command.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Create Series of Accounting Methods

Use the **aaa accounting** command to create default or named method lists defining specific accounting methods that can be used for each line or interface.

Currently, the software supports both the TACACS+ and RADIUS methods for accounting. The router reports user activity to the TACACS+ or RADIUS security server in the form of accounting records. Each accounting record contains accounting AV pairs and is stored on the security server.

Method lists for accounting define the way accounting is performed, enabling you to designate a particular security protocol to be used on specific lines or interfaces for particular types of accounting services. When naming a method list, do not use the names of methods, such as TACACS+.

For minimal accounting, include the **stop-only** keyword to send a "stop accounting" notice at the end of the requested user process. For more accounting, you can include the **start-stop** keyword, so that the external AAA server sends a "start accounting" notice at the beginning of the requested process and a "stop accounting" notice at the end of the process. In addition, you can use the **aaa accounting update** command to periodically send update records with accumulated information. Accounting records are stored only on the TACACS+ or RADIUS server.

When AAA accounting is activated, the router reports these attributes as accounting records, which are then stored in an accounting log on the security server.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 Do one of the following:

- **aaa accounting** {**commands** | **exec** | **network**} {**default** | *list-name*} {**start-stop** | **stop-only**}
- {none | method}

Example:

```
RP/0//CPU0:router(config)# aaa accounting commands default stop-only group tacacs+
```

Note
Command accounting is not supported on RADIUS, but supported on TACACS.

Creates a series of accounting methods, or a method list.

- The **commands** keyword enables accounting for XR EXEC mode shell commands.
- The **exec** keyword enables accounting for an interactive (XR EXEC mode) session.
- The **network** keyword enables accounting for all network-related service requests, such as Point-to-Point Protocol (PPP).
- The **default** keyword causes the listed accounting methods that follow this keyword to be the default list of methods for accounting.
- A *list-name* character string identifies the accounting method list.

- The **start-stop** keyword sends a “start accounting” notice at the beginning of a process and a “stop accounting” notice at the end of a process. The requested user process begins regardless of whether the “start accounting” notice was received by the accounting server.
- The **stop-only** keyword sends a “stop accounting” notice at the end of the requested user process.
- The **none** keyword states that no accounting is performed.
- The method list itself follows the **start-stop** keyword. Method list types are entered in the preferred sequence. The method argument lists the following types:
  - **group tacacs+**—Use the list of all configured TACACS+ servers for accounting.
  - **group radius—**Use the list of all configured RADIUS servers for accounting.
  - **group** *group-name*—Use a named server group, a subset of TACACS+ or RADIUS servers for accounting as defined by the **aaa group server tacacs+** or **aaa group server radius** command.
- The example defines a **default** command accounting method list, in which accounting services are provided by a TACACS+ security server, with a stop-only restriction.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Generate Interim Accounting Records

This task enables periodic interim accounting records to be sent to the accounting server. When the **aaa accounting update** command is activated, software issues interim accounting records for all users on the system.

Note Interim accounting records are generated only for network sessions, such as Internet Key Exchange (IKE) accounting, which is controlled by the **aaa accounting** command with the **network** keyword. System, command, or EXEC accounting sessions cannot have interim records generated.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **aaa accounting update** {**newinfo** | **periodic** *minutes*}

Example:

```
RP/0//CPU0:router(config)# aaa accounting update periodic 30
```

Enables periodic interim accounting records to be sent to the accounting server.

- If the **newinfo** keyword is used, interim accounting records are sent to the accounting server every time there is new accounting information to report. An example of this report would be when IPCP completes IP address negotiation with the remote peer. The interim accounting record includes the negotiated IP address used by the remote peer.
- When used with the **periodic** keyword, interim accounting records are sent periodically as defined by the argument number. The interim accounting record contains all the accounting information recorded for that user up to the time the interim accounting record is sent.

Caution
The **periodic** keyword causes heavy congestion when many users are logged in to the network.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Apply Method List

After you use the **aaa authorization** command to define a named authorization method list (or use the default method list) for a particular type of authorization, you must apply the defined lists to the appropriate lines in order for authorization to take place. Use the **authorization** command to apply the specified method lists (or, if none is specified, the default method list) to the selected line or group of lines.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **line** { **console** | **default** | **template** *template-name*}

Example:

```
RP/0//CPU0:router(config)# line console
```

Enters line template configuration mode.

Step 3 **authorization** {**commands** | **exec**} {**default** | *list-name*}

Example:

```
RP/0//CPU0:router(config-line)# authorization commands listname5
```

Enables AAA authorization for a specific line or group of lines.

- The **commands** keyword enables authorization on the selected lines for all commands.
- The **exec** keyword enables authorization for an interactive (XR EXEC mode) session.
- Enter the **default** keyword to apply the name of the default method list, as defined with the **aaa authorization** command.
- Enter the name of a list of authorization methods to use. If no list name is specified, the system uses the default. The list is created with the **aaa authorization** command.
- The example enables command authorization using the method list named listname5.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

After applying authorization method lists by enabling AAA authorization, apply accounting method lists by enabling AAA accounting.

Enable Accounting Services

This task enables accounting services for a specific line of group of lines.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **line { console | default | template template-name**}

Example:

```
RP/0//CPU0:router(config)# line console
```

Enters line template configuration mode.

Step 3 **accounting** {**commands** | **exec**} {**default** | *list-name*}

Example:

```
RP/0//CPU0:router(config-line)# accounting commands listname7
```

Enables AAA accounting for a specific line or group of lines.

- The **commands** keyword enables accounting on the selected lines for all XR EXEC mode shell commands.
- The **exec** keyword enables accounting for an interactive (XR EXEC mode) session.
- Enter the **default** keyword to apply the name of the default method list, as defined with the **aaa accounting** command.
- Enter the name of a list of accounting methods to use. If no list name is specified, the system uses the default. The list is created with the **aaa accounting** command.
- The example enables command accounting using the method list named listname7.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

After applying accounting method lists by enabling AAA accounting services, configure login parameters.

Configure Login Parameters

This task sets the interval that the server waits for reply to a login.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **line template** *template-name*

Example:

```
RP/0//CPU0:router(config)# line template alpha
```

Specifies a line to configure and enters line template configuration mode.

Step 3 **timeout login response** *seconds*

Example:

```
RP/0//CPU0:router(config-line)# timeout login response 20
```

Sets the interval that the server waits for reply to a login.

- The *seconds* argument specifies the timeout interval (in seconds) from 0 to 300. The default is 30 seconds.
- The example shows how to change the interval timer to 20 seconds.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Task Maps

For users who are authenticated using an external TACACS+ server and RADIUS server, Cisco IOS XR software AAA supports a method to define task IDs remotely.

Format of the Task String

The task string in the configuration file of the TACACS+ server consists of tokens delimited by a comma (,). Each token contains either a task ID name and its permissions or the user group to include for this particular user, as shown in the following example:

task = " *permissions* : *taskid name* , # *usergroup name* , ..."

Note Cisco IOS XR software allows you to specify task IDs as an attribute in the external RADIUS or TACACS+ server. If the server is also shared by non-Cisco IOS XR software systems, these attributes are marked as optional as indicated by the server documentation. For example, CiscoSecure ACS and the freeware TACACS+ server from Cisco require an asterisk (*) instead of an equal sign (=) before the attribute value for optional attributes. If you want to configure attributes as optional, refer to the TACACS+ server documentation.

For example, to give a user named user1 BGP read, write, and execute permissions and include user1 in a user group named operator, the username entry in the external server's TACACS+ configuration file would look similar to the following:

```
user = user1{
member = some-tac-server-group
opap = cleartext "lab"
service = exec {
task = "rwx:bgp,#operator"
}
}
```

The r,w,x, and d correspond to read, write, execute and debug, respectively, and the pound sign (#) indicates that a user group follows.

Note The optional keyword must be added in front of "task" to enable interoperability with systems based on Cisco IOS software.

If CiscoSecure ACS is used, perform the following procedure to specify the task ID and user groups:

Procedure

Step 1 Enter your username and password.

Step 2 Click the **Group Setup** button to display the **Group Setup** window.

Step 3 From the Group drop-down list, select the group that you want to update.

Step 4 Click the **Edit Settings** button.

Step 5 Use the scroll arrow to locate the Shell (exec) check box.

Step 6 Check the **Shell (exec)** check box to enable the custom attributes configuration.

Step 7 Check the **Custom attributes** check box.

Step 8 Enter the following task string without any blank spaces or quotation marks in the field:

Example:

```
task=rwx:bgp,#netadmin
```

Step 9 Click the **Submit + Restart** button to restart the server.

The following RADIUS Vendor-Specific Attribute (VSA) example shows that the user is part of the sysadmin predefined task group, can configure BGP, and can view the configuration for OSPF:

Example:

```
user Auth-Type := Local, User-Password == lab
        Service-Type = NAS-Prompt-User,
        Reply-Message = "Hello, %u",
        Login-Service = Telnet,
        Cisco-AVPair = "shell:tasks=#sysadmin,rwx:bgp,r:ospf"
```

After user1 successfully connects and logs in to the external TACACS+ server with username user1 and appropriate password, the **show user tasks** command can be used in XR EXEC mode to display all the tasks user1 can perform. For example:

Example:

```
Username:user1
Password:
RP/0/RP0/CPU0:router# show user tasks

Task:      basic-services  :READ    WRITE    EXECUTEDEBUG
Task:                 bgp  :READ    WRITE    EXECUTE
Task:                 cdp  :READ
Task:                diag  :READ
Task:          ext-access  :READ             EXECUTE
Task:             logging  :READ
```

Alternatively, if a user named user2, who does not have a task string, logs in to the external server, the following information is displayed:

Example:

```
Username:user2
Password:
RP/0/RP0/CPU0:router# show user tasks
No task ids available
```

How to Configure Hold-Down Timer for TACACS+

By default, the hold-down timer for TACACS+ is disabled. To enable the hold-down timer, use the **holddown-time** command under respective configuration modes as per the following hierarchy levels:

- **Global Level**: Applicable to all TACACS+ servers that are configured on the router.
- **Server Group Level**: Applicable only to TACACS+ servers that are configured in a particular server group. This configuration overrides the global hold-down timer configuration.
- **Server Level**: Applicable only to a particular TACACS+ server (that also includes the private server). This configuration overrides the timer value at all other levels.
- **Private Server Level**: Applicable only to a particular private TACACS+ server.

While selecting the timer at various configuration levels, the router gives preference to the one which is more specific to the server. That is, the server-level timer has the highest precedence, followed by server group-level and finally, the global-level timer.

Guidelines for Configuring Hold-Down Timer for TACACS+

- You must configure the TACACS+ servers for this feature to take effect.

- A timer value of zero indicates that the feature is disabled.
- The timer value is decided by the configuration that is closest to the server regardless of its value. That is, if the server-level timer is configured as 0, the system disables the feature for that particular server, even if a positive value exists at other levels. So, if you need to disable the feature for some servers or server-groups, and not for others, you can configure a zero value for those specific servers or server-groups, and configure a positive value at the global level.
- The system assigns priority to the servers based on the order in which they are configured in the router. The server that is configured first is used first. If the first server becomes unavailable or unreachable, the second server is used, and so on.
- Avoid configuring a large timer value, as it marks the server as being down for a longer period. Also, the router does not use that server for further client requests during the hold-down time, even if the server becomes available in between. As a result, we recommend that you configure an optimal timer value of say, one or two minutes.
- If there is a process restart or router reload while the timer is running, the timer immediately expires, and the router considers the unresponsive server as being up.

Syslog for Hold-Down Timer

The TACACS+ hold-down timer feature introduces a new syslog to notify that the server is marked as being down, and that the hold-down timer has started. This syslog replaces the old syslog which was invoked during earlier scenarios when server was down. If the feature is not enabled, the router continues to display the old syslog.

The syslog without enabling hold-down timer:

```
RP/0/RP0/CPU0:Aug 21 17:42:49.664 UTC: tacacsd[1226]: %SECURITY-TACACSD-6-SERVER_DOWN :
TACACS+ server 10.10.10.2/2020 is DOWN [vrf: 0x60000000, server-private: No]- Socket 116:
No route to host
```

The syslog with hold-down timer enabled:

```
RP/0/RP0/CPU0:ios#RP/0/RP0/CPU0:Aug 21 16:00:25.200 UTC: tacacsd[1227]:
%SECURITY-TACACSD-6-HOLDDOWN_TIME_START :
TACACS+ server 10.105.236.103/2020 is DOWN [vrf: 0x60000000, server-private: Yes]. Server
will be marked as DOWN for 20 seconds: Success
```

Configuration Example

- **Global Level**:

```
Router#configure
Router(config)#tacacs-server holddown-time 30
```

- **Server Level**:

```
Router(config)#tacacs-server host 10.105.236.102 port 2020
Router(config-tacacs-host)#holddown-time 35
```

- **Server-Group Level**:

```
Router#configure
```

```
Router(config)#aaa group server tacacs+ test-group
Router(config-sg-tacacs)#holddown-time 40
```

- **Private Server Level**:

```
Router(config)#aaa group server tacacs+ test-group
Router(config-sg-tacacs)#server-private 10.105.236.109 port 2020
Router(config-sg-tacacs-private)#holddown-time 55
```

Running Configuration

```
Router#show running-config
!
tacacs-server holddown-time 30
!
tacacs-server host 10.105.236.102 port 2020
 holddown-time 35
!
aaa group server tacacs+ test-group
 holddown-time 40
 server-private 10.105.236.109 port 2020
  holddown-time 55
 !
!
```

How to Disable Hold-Down Timer for TACACS+

You can disable the hold-down timer for TACACS+ at respective levels either by using the **no** form of the **holddown-time** command, or by configuring a timer value of zero.

For example,

```
Router(config)#no tacacs-server holddown-time 30
OR
Router(config)#tacacs-server holddown-time 0
```

Verification

A new field, **on-hold**, is introduced in the output field of the **show tacacs** command to indicate whether a server is on hold due to the hold-down timer or the server probe is in progress. A value of *true* indicates that the server is marked as being down. The router does not use that server for addressing any client request.

```
Router#show tacacs
Wed Oct 21 06:45:38.341 UTC
Server: 10.105.236.102/2020 opens=1 closes=1 aborts=1 errors=0
       packets in=0 packets out=0
       status=down single-connect=false family=IPv4
       idle-timeout=0 on-hold=true

Server: 10.105.236.103/2020 vrf=default [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4
        on-hold=true
```

The following is a sample output with **on-hold** value as *false*, which indicates that the server is not marked as being down. The router considers that server as being available for addressing client requests.

```
Router#show tacacs
Fri Aug 21 15:57:02.139 UTC

Server: 10.105.236.102/2020 opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4
        idle-timeout=0 on-hold=false

Server: 10.105.236.103/2020 vrf=default [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4
        on-hold=false
```

Related Topics

- Hold-Down Timer for TACACS+, on page 86

Associated Commands

- **holddown-time**

Overview on AAA Services

This section lists all the conceptual information that a software user must understand before configuring user groups and task groups through AAA or configuring Remote Authentication Dial-in User Service (RADIUS) or TACACS+ servers. Conceptual information also describes what AAA is and why it is important.

User, User Groups, and Task Groups

User attributes form the basis of the Cisco software administrative model. Each router user is associated with the following attributes:

- User ID (ASCII string) that identifies the user uniquely across an administrative domain
- Length limitation of 253 characters for passwords and one-way encrypted secrets
- List of user groups (at least one) of which the user is a member (thereby enabling attributes such as task IDs).

User Categories

Router users are classified into the following categories:

- Root Secure Domain Router (SDR) user (specific SDR administrative authority)
- SDR user (specific SDR user access)

Root System Users

The root system user is the entity authorized to “own” the entire router chassis. The root system user functions with the highest privileges over all router components and can monitor all secure domain routers in the system. At least one root system user account must be created during router setup. Multiple root system users can exist.

The root system user can perform any configuration or monitoring task, including the following:

- Configure secure domain routers.
- Create, delete, and modify root SDR users (after logging in to the secure domain router as the root system user).
- Create, delete, and modify secure domain router users and set user task permissions (after logging in to the secure domain router as the root system user).
- Access fabric racks or any router resource not allocated to a secure domain router, allowing the root system user to authenticate to any router node regardless of the secure domain router configurations.

Root SDR Users

A root SDR user controls the configuration and monitoring of a particular SDR. The root SDR user can create users and configure their privileges within the SDR. Multiple root SDR users can work independently. A single SDR may have more than one root SDR user.

A root SDR user can perform the following administrative tasks for a particular SDR:

- Create, delete, and modify secure domain router users and their privileges for the SDR.
- Create, delete, and modify user groups to allow access to the SDR.
- Manage nearly all aspects of the SDR.

A root SDR user cannot deny access to a root system user.

Secure Domain Router (SDR) Users

A SDR user has restricted access to an SDR as determined by the root SDR user. The SDR user performs the day-to-day system and network management activities. The tasks that the secure domain router user is allowed to perform are determined by the task IDs associated with the user groups to which the SDR user belongs. Multiple SDRs in a chasis are not supported.

User Groups

A *user group* defines a collection of users that share a set of attributes, such as access privileges. Cisco software allows the system administrator to configure groups of users and the job characteristics that are common in groups of users. Users are not assigned to groups by default hence the assignment needs to be done explicitly. A user can be assigned to more than one group.

Each user may be associated with one or more user groups. User groups have the following attributes:

- A user group consists of the list of task groups that define the authorization for the users. All tasks, except cisco-support, are permitted by default for root system users.
- Each user task can be assigned read, write, execute, or debug permission.

Predefined User Groups

The Cisco software provides a collection of user groups whose attributes are already defined. The predefined groups are as follows:

- **cisco-support:** This group is used by the Cisco support team.

- **maintenance:** Has the ability to display, configure and execute commands for network, files and user-related entities.
- **netadmin:** Has the ability to control and monitor all system and network parameters.
- **operator:** A demonstration group with basic privileges.
- **provisioning:** Has the ability to display and configure network, files and user-related entities.
- **read-only-tg:** Has the ability to perform any show command, but no configuration ability.
- **retrieve:** Has the ability to display network, files and user-related information.
- **root-lr:** Has the ability to control and monitor the specific secure domain router.
- **serviceadmin:** Service administration tasks, for example, Session Border Controller (SBC).
- **sysadmin:** Has the ability to control and monitor all system parameters but cannot configure network protocols.

To verify the individual permissions of a user group, assign the group to a user and execute the **show user tasks** command.

User-Defined User Groups

Administrators can configure their own user groups to meet particular needs.

User Group Inheritance

A user group can derive attributes from another user group. (Similarly, a task group can derive attributes from another task group). For example, when user group A inherits attributes from user group B, the new set of task attributes of the user group A is a union of A and B. The inheritance relationship among user groups is dynamic in the sense that if group A inherits attributes from group B, a change in group B affects group A, even if the group is not reinherited explicitly.

Task Groups

Task groups are defined by lists of permitted task IDs for each type of action (such as read, write, and so on). The task IDs are basically defined in the router system. Task ID definitions may have to be supported before task groups in external software can be configured.

Task IDs can also be configured in external TACACS+ or RADIUS servers.

Predefined Task Groups

The following predefined task groups are available for administrators to use, typically for initial configuration:

- **cisco-support:** Cisco support personnel tasks
- **netadmin:** Network administrator tasks
- **operator:** Operator day-to-day tasks (for demonstration purposes)
- **root-lr:** Secure domain router administrator tasks
- **sysadmin:** System administrator tasks
- **serviceadmin:** Service administration tasks, for example, SBC

User-Defined Task Groups

Users can configure their own task groups to meet particular needs.

Group Inheritance

Task groups support inheritance from other task groups. (Similarly, a user group can derive attributes from another user group. For example, when task group A inherits task group B, the new set of attributes of task group A is the union of A and B.

Command Access in XR and Admin Modes

The XR user group and task is mapped to the System Admin VM group when the System Admin mode is accessed from XR mode using **admin** command. The corresponding access permission on System Admin VM is available to the user. Currently, only aaa, admin task and root-lr groups are mapped to System Admin VM group or task. The other tasks like protocols are not mapped as these services are not supported in System Admin VM. The disaster-recovery user of System Admin VM is synced with the Host VM.

XR Task or Group	Sysadmin VM Group	Access	Example

root-lr	Root-system group	Full access to the system configuration.	`RP/0/RP0/CPU0:ios#show user group`
			`Mon Nov 3 13:48:54.536 UTC`
			`root-lr, cisco-support`
			`RP/0/RP0/CPU0:ios#show user tasks | inc root-lr`
			`Mon Nov 3 13:49:06.495 UTC`
			`Task: root-lr : READ WRITE EXECUTE DEBUG (reserved)`
			
			`RP/0/RP0/CPU0:ios#admin`
			`sysadmin-vm:0_RP0# show aaa user-group`
			`Mon Nov 3 13:48:00.790 UTC`
			`User group : root-system`

Admin-r/w/x/d	Admin-r	Read only commands on Sysadmin VM	`taskgroup tg-admin-write`
			` task write admin`
			` task execute admin`
			`!`
			`usergroup ug-admin-write`
			` taskgroup tg-admin-write`
			`!`
			`username admin-write`
			` group ug-admin-write`
			` password admin-write`
			`!`
			`RP/0/RP0/CPU0:ios#show user group`
			`Mon Nov 3 14:09:29.676 UTC`
			`ug-admin-write`
			`RP/0/RP0/CPU0:ios#show user tasks`
			`Mon Nov 3 14:09:35.244 UTC`
			`Task: admin : READ WRITE EXECUTE`
			
			`RP/0/RP0/CPU0:ios#admin`
			`Mon Nov 3 14:09:40.401 UTC`
			`admin-write connected from 127.0.0.1 using console on xr-vm_node0_RP0_CPU0`
			`sysadmin-vm:0_RP0# show aaa user-group Mon Nov 3 13:53:00.790 UTC User group : admin-r`

XR Task or Group	Sysadmin VM Group	Access	Example

Netadmin or sysadmin group
Admin-r/ wx /d, aaa -r/w/x/d
	Aaa -r and admin -r	Read only commands on Sysadmin VM	`RP/0/RP0/CPU0:ios#show user group`
			`Mon Nov 3 13:44:39.176 UTC`
			`netadmin`
			`RP/0/RP0/CPU0:ios#show user tasks | inc aaa`
			`Mon Nov 3 13:45:00.999 UTC`
			`Task: aaa : READ`
			`RP/0/RP0/CPU0:ios#show user tasks | inc admin`
			`Mon Nov 3 13:45:09.567 UTC`
			`Task: admin : READ`
			
			`RP/0/RP0/CPU0:ios#admin`
			`Mon Nov 3 13:46:21.183 UTC`
			`netadmin connected from 127.0.0.1 using console on xr-vm_node0_RP0_CPU0`
			`sysadmin-vm:0_RP0# show aaa user-group`
			`Mon Nov 3 13:44:23.939 UTC`
			`User group : admin-r,aaa-r`
			`sysadmin-vm:0_RP0#`

Admin Access for NETCONF and gRPC Sessions

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Admin Access for NETCONF and gRPC Sessions	Release 7.4.1	This feature allows all authorized users on XR VM to access administration data on the router through NETCONF or gRPC interface, similar to accessing the CLI. This functionality works by internally mapping the task group of the user on XR VM to a predefined group on System Admin VM. Therefore, the NETCONF and gRPC users can access the admin-related information on the router even if their user profiles do not exist on System Admin VM. Prior to this release, only those users who were authorized on XR VM could access System Admin VM through CLI, by using the **admin** command. Users that were not configured on System Admin VM were denied access through the NETCONF or gRPC interfaces.

NETCONF is an XML-based protocol used over Secure Shell (SSH) transport to configure a network. Similarly, gRPC is an open-source remote procedure call framework. The client applications can use these protocols to

request information from the router and make configuration changes to the router. Prior to Cisco IOS XR Software Release 7.4.1, users who use NETCONF, gRPC or any other configuration interface, other than CLI, to access the admin-related information on the router, had to belong to user groups that are configured on System Admin VM. Otherwise, the router would issue an UNAUTHORIZED access error message and deny access through that client interface.

By default, XR VM synchronizes only the first-configured user to System Admin VM. If you delete the first-user in XR VM, the system synchronizes the next user in the **root-lr** group (which is the highest privilege group in XR VM for Cisco IOS XR 64-bit platforms) to System Admin VM only if there are no other users configured in System Admin VM. The system does not automatically synchronize the subsequent users to System Admin VM. Therefore, in earlier releases, users whose profiles did not exist in System Admin VM were not able to perform any NETCONF or gRPC operations on System Admin VM.

From Cisco IOS XR Software Release 7.4.1 and later, the system internally maps the users who are authorized on XR VM to System Admin VM of the router, based on the task table of the user on XR VM. With this feature, the NETCONF and gRPC users can access admin-related information on the router even if their user profiles do not exist on System Admin VM. By default, this feature is enabled.

To know more about NETCONF and gRPC operations, see the *Use NETCONF Protocol to Define Network Operations with Data Models* chapter and *Use gRPC Protocol to Define Network Operations with Data Models* chapter in the *Programmability Configuration Guide*.

User Profile Mapping from XR VM to System Admin VM

User privileges to execute commands and access data elements on the router are usually specified using certain command rules and data rules that are created and applied on the user groups.

For details on user groups, command rules and data rules, see the *Create User Profiles and Assign Privilege* chapter in the *System Setup and Software Installation Guide for Cisco NCS 560 Series Routers*.

When the internal process for AAA starts or when you create the first user, the system creates the following set of predefined groups, command rules and data rules in System Admin VM. These configurations are prepopulated to allow users of different groups (such as **root-system**, **admin-r** and **aaa-r**) in System Admin VM.

You can use the **show running-configuration aaa** command to view the AAA configurations.

```
aaa authentication groups group aaa-r gid 100 users %%__system_user__%%
!
aaa authentication groups group admin-r gid 100 users %%__system_user__%%
!
aaa authentication groups group root-system gid 100 users "%%__system_user__%% "
!
aaa authorization cmdrules cmdrule 1 context * command * group root-system ops rx action
accept
!
aaa authorization cmdrules cmdrule 2 context * command "show running-config aaa" group aaa-r
 ops rx action accept
!
aaa authorization cmdrules cmdrule 3 context * command "show tech-support aaa" group aaa-r
 ops rx action accept
!
aaa authorization cmdrules cmdrule 4 context * command "show aaa" group   aaa-r ops rx
action accept
!
aaa authorization cmdrules cmdrule 5 context * command show group   admin-r ops rx action
accept
!
aaa authorization datarules datarule 1 namespace * context * keypath * group root-system
```

```
ops rwx action accept
!
aaa authorization datarules datarule 2 namespace * context * keypath /aaa group aaa-r ops
r action accept
!
aaa authorization datarules datarule 3 namespace * context * keypath /aaa group admin-r ops
 rwx action reject
!
aaa authorization datarules datarule 4 namespace * context * keypath / group admin-r ops r
 action accept
!
```

The admin CLI for the user works based on the above configurations. The **root-system** is the group with the highest privilege in System Admin VM. The **admin-r** group has only read and execute access to all data. The **aaa-r** group has access only to AAA data. With the introduction of the admin access feature for all users, the NETCONF and gRPC applications can also access the admin data based on the above rules and groups.

User Profile Mapping Based on Task-ID

This table shows the internal mapping of XR VM users to System Admin VM. The users in XR VM belong to various user groups such as **aaa**, **admin**, **root-lr** and **root-system**.

XR VM User Group:Task-ID	System Admin VM User Group
aaa:rwxd	**aaa-r**
aaa:rwx	**aaa-r**
aaa:rw	**aaa-r**
aaa:wx	**aaa-r**
aaa:rx	**aaa-r**
aaa:r	**aaa-r**
aaa:w	**aaa-x**
aaa:x	**aaa-x**
root-system:rwxd	**root-system**
root-lr:rwxd	**root-system**
admin:rwxd	**admin-r**
admin:rwx	**admin-r**
admin:rw	**admin-r**
admin:r	**admin-r**

How to Allow Read Access to Administration Data for NETCONF and gRPC Clients

NETCONF and gRPC users access the administration data on the router through GET operations as defined by the respective protocols. To allow this read access to administration data for users belonging to **admin-r** group, you must configure a new command rule specifically for the NETCONF or gRPC client.

Configuration Example

```
Router#admin
sysadmin-vm:0_RP0#configure
sysadmin-vm:0_RP0(config)#aaa authorization cmdrules cmdrule 6
sysadmin-vm:0_RP0(config-cmdrule-6)#context netconf
sysadmin-vm:0_RP0(config-cmdrule-6)#command get
sysadmin-vm:0_RP0(config-cmdrule-6)#group admin-r
sysadmin-vm:0_RP0(config-cmdrule-6)#ops rx
sysadmin-vm:0_RP0(config-cmdrule-6)#action accept
sysadmin-vm:0_RP0(config)#commit
```

Running Configuration

```
aaa authorization cmdrules cmdrule 6
 context netconf
 command get
 group   admin-r
 ops     rx
 action  accept
!
```

Associated Command

- **aaa authorization (System Admin-VM)**

Administrative Model

The router operates in two planes: the administration (admin) plane and secure domain router (SDR) plane. The admin (shared) plane consists of resources shared across all SDRs, while the SDR plane consists of those resources specific to the particular SDR.

Each SDR has its own AAA configuration including, local users, groups, and TACACS+ and RADIUS configurations. Users created in one SDR cannot access other SDRs unless those same users are configured in the other SDRs.

Administrative Access

Administrative access to the system can be lost if the following operations are not well understood and carefully planned.

- Configuring authentication that uses remote AAA servers that are not available, particularly authentication for the console.

Note The **none** option without any other method list is not supported.

- Configuring command authorization or XR EXEC mode authorization on the console should be done with extreme care, because TACACS+ servers may not be available or may deny every command, which locks the user out. This lockout can occur particularly if the authentication was done with a user not known to the TACACS+ server, or if the TACACS+ user has most or all the commands denied for one reason or another.

To avoid a lockout, we recommend these:

- Before turning on TACACS+ command authorization or XR EXEC mode authorization on the console, make sure that the user who is configuring the authorization is logged in using the appropriate user permissions in the TACACS+ profile.
- If the security policy of the site permits it, use the **none** option for command authorization or XR EXEC mode authorization so that if the TACACS+ servers are not reachable, AAA rolls over to the **none** method, which permits the user to run the command.
- Make sure to allow local fallback when configuring AAA. See, Create Series of Authorization Methods, on page 48.
- If you prefer to commit the configuration on a trial basis for a specified time, you may do so by using the **commit confirmed** command, instead of direct **commit**.

AAA Database

The AAA database stores the users, groups, and task information that controls access to the system. The AAA database can be either local or remote. The database that is used for a specific situation depends on the AAA configuration.

Local Database

AAA data, such as users, user groups, and task groups, can be stored locally within a secure domain router. The data is stored in the in-memory database and persists in the configuration file. The stored passwords are encrypted.

Note The database is local to the specific secure domain router (SDR) in which it is stored, and the defined users or groups are not visible to other SDRs in the same system.

You can delete the last remaining user from the local database. If all users are deleted when the next user logs in, the setup dialog appears and prompts you for a new username and password.

Note The setup dialog appears only when the user logs into the console.

Remote Database

AAA data can be stored in an external security server, such as CiscoSecure ACS. Security data stored in the server can be used by any client (such as a network access server [NAS]) provided that the client knows the server IP address and shared secret.

Remote AAA Configuration

Products such as CiscoSecure ACS can be used to administer the shared or external AAA database. The router communicates with the remote AAA server using a standard IP-based security protocol (such as TACACS+ or RADIUS).

Client Configuration

The security server should be configured with the secret key shared with the router and the IP addresses of the clients.

User Groups

User groups that are created in an external server are not related to the user group concept that is used in the context of local AAA database configuration on the router. The management of external TACACS+ server or RADIUS server user groups is independent, and the router does not recognize the user group structure. The remote user or group profiles may contain attributes that specify the groups (defined on the router) to which a user or users belong, as well as individual task IDs.

Configuration of user groups in external servers comes under the design of individual server products. See the appropriate server product documentation.

Task Groups

Task groups are defined by lists of permitted task IDs for each type of action (such as read, write, and so on). The task IDs are basically defined in the router system. Task ID definitions may have to be supported before task groups in external software can be configured.

Task IDs can also be configured in external TACACS+ or RADIUS servers.

AAA Configuration

This section provides information about AAA configuration.

Method Lists

AAA data may be stored in a variety of data sources. AAA configuration uses *method lists* to define an order of preference for the source of AAA data. AAA may define more than one method list and applications (such as login) can choose one of them. For example, console ports may use one method list and the vty ports may use another. If a method list is not specified, the application tries to use a default method list. If a default method list does not exist, AAA uses the local database as the source.

Rollover Mechanism

AAA can be configured to use a prioritized list of database options. If the system is unable to use a database, it automatically rolls over to the next database on the list. If the authentication, authorization, or accounting request is rejected by any database, the rollover does not occur and the request is rejected.

The following methods are available:

- Local: Use the locally configured database (not applicable for accounting and certain types of authorization)
- TACACS+: Use a TACACS+ server (such as CiscoSecure ACS)
- RADIUS: Use a RADIUS server
- Line: Use a line password and user group (applicable only for authentication)
- None: Allow the request (not applicable for authentication)



Note If the system rejects the authorization request and the user gets locked out, you can try to rollback the previous configuration or remove the problematic AAA configuration through auxiliary port. To log in to the auxiliary port, use the local username and password; not the tacacs+ server credentials. The **config_rollback -n 0x1** command can be used to rollback the previous configuration. If you are not able to access the auxiliary port, a router reload might be required in such scenarios.

Server Grouping

Instead of maintaining a single global list of servers, the user can form server groups for different AAA protocols (such as RADIUS and TACACS+) and associate them with AAA applications (such as PPP and XR EXEC mode).



Note In scenarios where multiple servers are within a TACACS AAA server group and multiple such groups exist within a remote database, the fallback behavior is when one server within a group is unreachable, and the system will default to the next server in that same group. However, when there is a mismatch in the shared secret between the router and a TACACS server, the router will not attempt to connect to the subsequent server in the group. Instead, it will bypass that group entirely and proceed to the next available method (group or local or none) based on the configuration.

Authentication

Authentication is the most important security process by which a principal (a user or an application) obtains access to the system. The principal is identified by a username (or user ID) that is unique across an administrative domain. The applications serving the user (such as or Management Agent) procure the username and the credentials from the user. AAA performs the authentication based on the username and credentials passed to it by the applications. The role of an authenticated user is determined by the group (or groups) to which the user belongs. (A user can be a member of one or more user groups.)

Authentication of Non-Owner Secure Domain Router User

When logging in from a non-owner secure domain router, the root system user must add the "@admin" suffix to the username. Using the "@admin" suffix sends the authentication request to the owner secure domain router for verification. The owner secure domain router uses the methods in the list-name **remote** for choosing the authentication method. The **remote** method list is configured using the **aaa authentication login remote** *method1 method2...* command.

Authentication of Owner Secure Domain Router User

An owner secure domain router user can log in only to the nodes belonging to the specific secure domain router associated with that owner secure domain router user. If the user is member of a root-sdr group, the user is authenticated as an owner secure domain router user.

Authentication of Secure Domain Router User

Secure domain router user authentication is similar to owner secure domain router user authentication. If the user is not found to be a member of the designated owner secure domain router user group, the user is authenticated as a secure domain router user.

Authentication Flow of Control

AAA performs authentication according to the following process:

1. A user requests authentication by providing a username and password (or secret).
2. AAA verifies the user's password and rejects the user if the password does not match what is in the database.
3. AAA determines the role of the user (root SDR user, or SDR user).

- If the user has been configured as a member of an owner secure domain router user group, then AAA authenticates the user as an owner secure domain router user.
- If the user has not been configured as a member of an owner secure domain router user group, AAA authenticates the user as a secure domain router user.

Clients can obtain a user's permitted task IDs during authentication. This information is obtained by forming a union of all task group definitions specified in the user groups to which the user belongs. Clients using such information typically create a session for the user (such as an API session) in which the task ID set remains static. Both the XR EXEC mode and external API clients can use this feature to optimize their operations. XR EXEC mode can avoid displaying the commands that are not applicable and an EMS application can, for example, disable graphical user interface (GUI) menus that are not applicable.

If the attributes of a user, such as user group membership and, consequently, task permissions, are modified, those modified attributes are not reflected in the user's current active session; they take effect in the user's next session.

Password Types

In configuring a user and that user's group membership, you can specify two types of passwords: encrypted or clear text.

The router supports both two-way and one-way (secret) encrypted user passwords. Secret passwords are ideal for user login accounts because the original unencrypted password string cannot be deduced on the basis of the encrypted secret. Some applications (PPP, for example) require only two-way passwords because they must decrypt the stored password for their own function, such as sending the password in a packet. For a login user, both types of passwords may be configured, but a warning message is displayed if one type of password is configured while the other is already present.

If both secret and password are configured for a user, the secret takes precedence for all operations that do not require a password that can be decrypted, such as login. For applications such as PPP, the two-way encrypted password is used even if a secret is present.

Type 8 and Type 9 Passwords

This feature provides the options for Type 8 and Type 9 passwords in AAA security services. The Type 8 and Type 9 passwords provide more secure and robust support for saving passwords w.r.t each username. Thus, in scenarios where a lot of confidential data need to be maintained, these encryption methods ensure that the admin and other user passwords are strongly protected.

The implementation of Type 8 password uses SHA256 hashing algorithm, and the Type 9 password uses scrypt hashing algorithm.



Note The Type 8 and Type 9 passwords are supported on the IOS XR 64-bit operating system starting from Cisco IOS XR Software Release 7.0.1.

Type 10 Password

The Cisco IOS XR 64-bit software introduces the support for Type 10 password that uses **SHA512** encryption algorithm. The **SHA512** encryption algorithm provides improved security to the user passwords compared to the older algorithms such as **MD5** and **SHA256**. With this feature, **SHA512** becomes the default encryption algorithm for the passwords in user name configuration, even for the first user creation scenario. Prior to the introduction of Type 10 password, **MD5** was used as the default algorithm.

To configure Type 10 password, see Configure Type 10 Password.

Restrictions for Type 10 Password Usage

These restrictions apply to the usage of Type 10 password:

- Backward compatibility issues such as configuration loss, authentication failure, and so on, are expected when you downgrade to lower versions that still use **MD5** or **SHA256** encryption algorithms. Convert the passwords to Type 10 before such downgrades to minimize the impact of such issues. For details, see Backward Compatibility for Password Types, on page 16.
- In a first user configuration scenario or when you reconfigure a user, the system syncs only the Type 5 and Type 10 passwords from XR VM to System Admin VM and Host VM. It doesn't sync the Type 8 and Type 9 passwords in such scenarios.

AAA Password Security for FIPS Compliance

Cisco IOS XR Software introduces advanced AAA password strengthening policy and security mechanism to store, retrieve and provide rules or policy to specify user passwords. This password policy is applicable only for local users, and not for remote users whose profile information are stored in a third party AAA server. This policy is not applicable to secrets of the user. If both secret and password are configured for a user, then secret takes precedence, and password security policy does not have any effect on authentication or change of password for such users. This AAA password security policy works as such for Cisco IOS XR platforms. Whereas, this feature is supported only on XR VM, for Cisco IOS XR 64 bit platforms.

High Availability for AAA Password Security Policy

The AAA password policy configurations and username configurations remain intact across RP failovers or process restarts in the system. The operational data such as, lifetime of the password and lockout time of the user are not stored on system database or disk. Hence, those are not restored across RP failovers or process restarts. Users start afresh on the active RP or on the new process. Hence, users who were locked out before RP failover or process restart are able to login immediately after the failover or restart.

To configure AAA password policy, see Configure AAA Password Policy, on page 16.

AAA Password Security Policies

AAA password security for FIPS compliance consists of these policies:

Password Composition Policy

Passwords can be composed by any combination of upper and lower case alphabets, numbers and special characters that include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")". Security administrator can also set the types and number of required characters that comprise the password, thereby providing more flexibility for password composition rules. The minimum number of character change required between passwords is 4, by default. There is no restriction on the upper limit of the number of uppercase, lowercase, numeric and special characters.

Password Length Policy

The administrator can set the minimum and maximum length of the password. The minimum configurable length in password policy is 2, and the maximum length is 253.

Password Lifetime Policy

The administrator can configure a maximum lifetime for the password, the value of which can be specified in years, months, days, hours, minutes and seconds. The configured password never expires if this parameter is not configured. The configuration remains intact even after a system reload. But, the password creation time is updated to the new time whenever the system reboots. For example, if a password is configured with a life time of one month, and if the system reboots on 29th day, then the password is valid for one more month after the system reboot. Once the configured lifetime expires, further action is taken based on the password expiry policy (see the section on Password Expiry Policy).

Password Expiry Policy

If the password credential of a user who is trying to login is already expired, then the following actions occur:

- User is prompted to set the new password after successfully entering the expired password.
- The new password is validated against the password security policy.
- If the new password matches the password security policy, then the AAA data base is updated and authentication is done with the new password.
- If the new password is not compliant with the password security policy, then the attempt is considered as an authentication failure and the user is prompted again to enter a new password. The max limit for such attempts is in the control of login clients and AAA does not have any restrictions for that.

As part of password expiry policy, if the life time is not yet configured for a user who has already logged in, and if the security administrator configures the life time for the same user, then the life time is set in the database. The system checks for password expiry on the subsequent authentication of the same user.

Password expiry is checked only during the authentication phase. If the password expires after the user is authenticated and logged in to the system, then no action is taken. The user is prompted to change the password only during the next authentication of the same user.

Debug logs and syslog are printed for the user password expiry only when the user attempts to login. This is a sample syslog in the case of password expiry:

```
RP/0/RSP1/CPU0:Jun 21 09:13:34.241 : locald_DSC[308]: %SECURITY-LOCALD-5-USER_PASSWD_EXPIRED
 :
Password for user 'user12' has expired.
```

Password Change Policy

Users cannot change passwords at will. A password change is triggered in these scenarios:

- When the security administrator needs to change the password
- When the user is trying to get authenticated using a profile and the password for the profile is expired
- When the security administrator modifies the password policy which is associated to the user, and does not immediately change the password according to the policy

You can use the **show configuration failed** command to display the error messages when the password entered does not comply with the password policy configurations.

When the security administrator changes the password security policy, and if the existing profile does not meet the password security policy rules, no action is taken if the user has already logged in to the system. In

this scenario, the user is prompted to change the password when he tries to get authenticated using the profile which does not meet the password security rules.

When the user is changing the password, the lifetime of the new password remains same as that of the lifetime that was set by the security administrator for the old profile.

When password expires for non-interactive clients (such as dot1x), an appropriate error message is sent to the clients. Clients must contact the security administrator to renew the password in such scenarios.

Service Provision after Authentication

The basic AAA local authentication feature ensures that no service is performed before a user is authenticated.

User Re-authentication Policy

A user is re-authenticated when he changes the password. When a user changes his password on expiry, he is authenticated with the new password. In this case, the actual authentication happens based on the previous credential, and the new password is updated in the database.

User Authentication Lockout Policy

AAA provides a configuration option, **authen-max-attempts**, to restrict users who try to authenticate using invalid login credentials. This option sets the maximum number of permissible authentication failure attempts for a user. The user gets locked out when he exceeds this maximum limit, until the lockout timer (**lockout-time**) is expired. If the user attempts to login in spite of being locked out, the lockout expiry time keep advancing forward from the time login was last attempted.

This is a sample syslog when user is locked out:

```
RP/0/RSP1/CPU0:Jun 21 09:21:28.226 : locald_DSC[308]: %SECURITY-LOCALD-5-USER_PASSWD_LOCKED
 :
User 'user12’ is temporarily locked out for exceeding maximum unsuccessful logins.
```

This is a sample syslog when user is unlocked for authentication:

```
RP/0/RSP1/CPU0:Jun 21 09:14:24.633 : locald_DSC[308]: %SECURITY-LOCALD-5-USER_PASSWD_UNLOCKED
 :
User 'user12' is unlocked for authentications.
```

Password Policy Creation, Modification and Deletion

Security administrators having write permission for AAA tasks are allowed to create password policy. Modification is allowed at any point of time, even when the policy is associated to a user. Deletion of password policy is not allowed until the policy is un-configured from the user.

After the modification of password policy associated with a user, security administrator can decide if he wants to change passwords of associated users complying to the password policy. Based on this, there are two scenarios:

- If the administrator configures the password, then the user is not prompted to change the password on next login.
- If the administrator does not configure the password, then the user is prompted to change the password on next login.

In either of the above cases, at every password expiry interval, the user is prompted to change the password on next login.

Debug messages are printed when password policies are created, modified and deleted.

Minimum Password Length for First User Creation

To authenticate the user for the first time, Cisco router prompts you to create a username and password, in any of the following situations:

- When the Cisco Router is booted for the very first time.
- When the router is reloaded with no username configuration.
- When the already existing username configurations are deleted.

By default, the minimum length for passwords in a Cisco router is limited to two characters. Due to noise on the console, there is a possibility of the router being blocked out. Therefore, the minimum length for password has been increased to six characters for a first user created on the box, in each of the situations described above. This reduces the probability of the router being blocked out. It avoids the security risks that are caused due to very small password length. For all other users created after the first one, the default minimum length for password is still two characters.

For more information about how to configure a first user, see Configure First User on Cisco Routers, on page 9.

Password Policy for User Secret

The Cisco IOS XR Software extends the existing password policy support for the user authentication to all types of user secret. The types of secret include Type 5 (**MD5**), 8 (**SHA256**), 9 (**sCrypt**) and 10 (**SHA512**). Prior to this release, the support for password policy was only for the Type 7 passwords. The new policy is common to both password and secret of the user. Using irreversible hashed-secrets has the benefit that the other modules in the device cannot retrieve the clear-text form of these secrets. Thus, the enhancement provides more secure secrets for the user names. This policy for user secrets is applicable for local and remote users.

The classic Cisco IOS XR platforms support the password policy for secrets on the XR and the Admin plane. Whereas, the 64-bit Cisco IOS XR platforms support this feature only on XR VM; not on System Admin VM.

To configure password policy for user secret, see Configure Password Policy for User Secret and Password, on page 18.

Task-based Authorization

AAA employs “task permissions” for any control, configure, or monitor operation through CLI or API. The Cisco IOS software concept of privilege levels has been replaced in software by a task-based authorization system.

Task IDs

The operational tasks that enable users to control, configure, and monitor Cisco software are represented by task IDs. A task ID defines the permission to run an operation for a command. Users are associated with sets of task IDs that define the breadth of their authorized access to the router.

Task IDs are assigned to users through the following means:

Each user is associated with one or more user groups. Every user group is associated with one or more *task groups*; in turn, every task group is defined by a set of task IDs. Consequently, a user’s association with a

particular user group links that user to a particular set of task IDs. A user that is associated with a task ID can execute any operation associated with that task ID.

General Usage Guidelines for Task IDs

Most router control, configuration, or monitoring operation (CLI, Netconf, Restconf, XML API) is associated with a particular set of task IDs. Typically, a given CLI command or API invocation is associated with at least one or more task IDs. Neither the **config** nor the **commit** commands require any specific task id permissions. The configuration and commit operations do not require specific task ID permissions. Aliases also don't require any task ID permissions. You cannnot perform a configuration replace unless root-lr permissions are assigned. If you want to deny getting into configuration mode you can use the TACACS+ command authorization to deny the config command. These associations are hard-coded within the router and may not be modified. Task IDs grant permission to perform certain tasks; task IDs do not deny permission to perform tasks. Task ID operations can be one, all, or a combination of classes that are listed in this table.

Note Restconf will be supported in a future release.

Table 5: Task ID Classes

Operation	Description
Read	Specifies a designation that permits only a read operation.
Write	Specifies a designation that permits a change operation and implicitly allows a read operation.
Execute	Specifies a designation that permits an access operation; for example ping and Telnet.
Debug	Specifies a designation that permits a debug operation.

The system verifies that each CLI command and API invocation conforms with the task ID permission list for the user. If you are experiencing problems using a CLI command, contact your system administrator.

Multiple task ID operations separated by a slash (for example read/write) mean that both operations are applied to the specified task ID.

Multiple task ID operations separated by a comma (for example read/write, execute) mean that both operations are applied to the respective task IDs. For example, the **copy ipv4 access-list** command can have the read and write operations applied to the acl task ID, and the execute operation applied to the *filesystem* task ID.

If the task ID and operations columns have no value specified, the command is used without any previous association to a task ID and operation. In addition, users do not have to be associated to task IDs to use ROM monitor commands.

Users may need to be associated to additional task IDs to use a command if the command is used in a specific configuration submode. For example, to execute the **show redundancy** command, a user needs to be associated to the system (read) task ID and operations as shown in the following example:

```
RP/0/RP0/CPU0:router# show redundancy
```

Task IDs for TACACS+ and RADIUS Authenticated Users

Cisco software AAA provides the following means of assigning task permissions for users authenticated with the TACACS+ and RADIUS methods:

- Specify the text version of the task map directly in the configuration file of the external TACACS+ and RADIUS servers.
- Specify the privilege level in the configuration file of the external TACACS+ and RADIUS servers.
- Create a local user with the same username as the user authenticating with the TACACS+ and RADIUS methods.
- Specify, by configuration, a default task group whose permissions are applied to any user authenticating with the TACACS+ and RADIUS methods.

Privilege Level Mapping

For compatibility with TACACS+ daemons that do not support the concept of task IDs, AAA supports a mapping between privilege levels defined for the user in the external TACACS+ server configuration file and local user groups. Following TACACS+ authentication, the task map of the user group that has been mapped from the privilege level returned from the external TACACS+ server is assigned to the user. For example, if a privilege level of 5 is returned from the external TACACS server, AAA attempts to get the task map of the local user group priv5. This mapping process is similar for other privilege levels from 1 to 13. For privilege level 14 maps to the user group owner-sdr.

For example, with the Cisco freeware tac plus server, the configuration file has to specify *priv_lvl* in its configuration file, as shown in the following example:

```
user = sampleuser1{
    member = bar
    service = exec-ext {
        priv_lvl = 5
    }
}
```

The number 5 in this example can be replaced with any privilege level that has to be assigned to the user *sampleuser.*

XML Schema for AAA Services

The extensible markup language (XML) interface uses requests and responses in XML document format to configure and monitor AAA. The AAA components publish the XML schema corresponding to the content and structure of the data used for configuration and monitoring. The XML tools and applications use the schema to communicate to the XML agent for performing the configuration.

The following schema are published by AAA:

- Authentication, Authorization and Accounting configuration
- User, user group, and task group configuration
- TACACS+ server and server group configuration
- RADIUS server and server group configuration

Netconf and Restconf for AAA Services

Just as in XML schemas, in Netconf and Restconf, username and password is controlled by either local or triple A (AAA) services.

Note Restconf will be supported in a future release.

About RADIUS

RADIUS is a distributed client/server system that secures networks against unauthorized access. In the Cisco implementation, RADIUS clients run on Cisco routers and send authentication and accounting requests to a central RADIUS server that contains all user authentication and network service access information.

RADIUS is a fully open protocol, distributed in source code format, that can be modified to work with any security system currently available on the market.

Cisco supports RADIUS under its AAA security paradigm. RADIUS can be used with other AAA security protocols, such as TACACS+, Kerberos, and local username lookup.

Note RADIUS is supported on all Cisco platforms, but some RADIUS-supported features run only on specified platforms.

RADIUS has been implemented in a variety of network environments that require high levels of security while maintaining network access for remote users.

Use RADIUS in the following network environments that require access security:

- Networks with multiple-vendor access servers, each supporting RADIUS. For example, access servers from several vendors use a single RADIUS server-based security database. In an IP-based network with multiple vendors' access servers, dial-in users are authenticated through a RADIUS server that has been customized to work with the Kerberos security system.
- Turnkey network security environments in which applications support the RADIUS protocol, such as in an access environment that uses a "smart card" access control system. In one case, RADIUS has been used with Enigma security cards to validate users and grant access to network resources.
- Networks already using RADIUS. You can add a Cisco router with RADIUS to the network. This might be the first step when you make a transition to a Terminal Access Controller Access Control System Plus (TACACS+) server.
- Networks in which a user must access only a single service. Using RADIUS, you can control user access to a single host, utility such as Telnet, or protocol such as Point-to-Point Protocol (PPP). For example, when a user logs in, RADIUS identifies this user as having authorization to run PPP using IP address 10.2.3.4 and the defined access list is started.
- Networks that require resource accounting. You can use RADIUS accounting independent of RADIUS authentication or authorization. The RADIUS accounting functions allow data to be sent at the start and end of services, indicating the amount of resources (such as time, packets, bytes, and so on) used during the session. An Internet service provider (ISP) might use a freeware-based version of RADIUS access control and accounting software to meet special security and billing needs.
- Networks that support preauthentication. Using the RADIUS server in your network, you can configure AAA preauthentication and set up the preauthentication profiles. Preauthentication enables service providers to better manage ports using their existing RADIUS solutions and to efficiently manage the use of shared resources to offer differing service-level agreements.

Network Security Situations in Which RADIUS is Unsuitable

RADIUS is not suitable in the following network security situations:

- Multiprotocol access environments. RADIUS does not support the following protocols:
  - NetBIOS Frame Control Protocol (NBFCP)
  - NetWare Asynchronous Services Interface (NASI)
  - X.25 PAD connections
- Router-to-router situations. RADIUS does not provide two-way authentication. RADIUS can be used to authenticate from one router to a router other than a Cisco router if that router requires RADIUS authentication.
- Networks using a variety of services. RADIUS generally binds a user to one service model.

RADIUS Operation

When a user attempts to log in and authenticate to an access server using RADIUS, the following steps occur:

1. The user is prompted for and enters a username and password.
2. The username and encrypted password are sent over the network to the RADIUS server.
3. The user receives one of the following responses from the RADIUS server:
   a. ACCEPT—The user is authenticated.
   a. REJECT—The user is not authenticated and is prompted to reenter the username and password, or access is denied.
   a. CHALLENGE—A challenge is issued by the RADIUS server. The challenge collects additional data from the user.
   a. CHANGE PASSWORD—A request is issued by the RADIUS server, asking the user to select a new password.

   The ACCEPT or REJECT response is bundled with additional data used for XR EXEC mode or network authorization. You must first complete RADIUS authentication before using RADIUS authorization. The additional data included with the ACCEPT or REJECT packets consists of the following:

- Services that the user can access, including Telnet, rlogin, or local-area transport (LAT) connections, and PPP, Serial Line Internet Protocol (SLIP), or XR EXEC mode services.
- Connection parameters, including the host or client IP address, access list, and user timeouts.

RADIUS with TLS protection

Table 6: Feature History Table

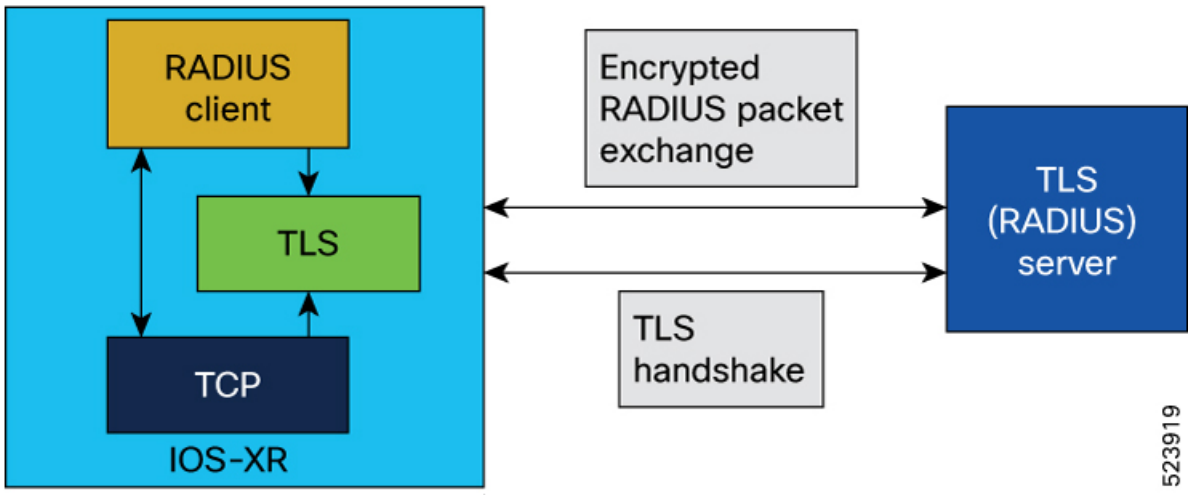
Feature Name	Release Information	Feature Description

RADIUS with TLS protection	Release 24.4.1	Remote Authentication Dial-In User Service (RADIUS) packets are now less vulnerable to security risks, including data exposure, replay attacks, weak authentication, and encryption weaknesses. This is because we have enabled support for RADIUS with TLS protection.
		
		You can configure the RADIUS protocol on the router to redirect RADIUS packets to a remote server over TLS for Authentication, Authorization, and Accounting (AAA) services.
		
		The feature introduces these changes:
		
		CLI:
		
		• The keyword **radsec-server** is introduced in the **radius-server host** command.
		
		YANG Data Models:
		
		• New Xpath for
		
		`Cisco-IOS-XR-um-aaa-cfg.yang`
		
		• New Xpath for
		
		`Cisco-IOS-XR-aaa-lib-cfg.yang`
		
		(see GitHub, YANG Data Models Navigator)

Topology for RADIUS with TLS protection

Traditionally, RADIUS has been used for Authentication, Authorization, and Accounting (AAA). However, to meet modern security demands it is important to enhance its encryption and authentication. To increase the resilience against threats and maintain a secure network environment, TLS is now utilized as the transport protocol for RADIUS.

This feature supports TLS version 1.3.



Let's understand how you can establish RADIUS communication with TLS.

Establish TLS session: The RADIUS client initiates the process to establish a secure TLS session with the RADIUS server, which acts as the TLS server. This session is built upon a TCP socket that the RADIUS client creates.

Store TLS context: Upon successful TLS connection, the TLS context is preserved within the TLS connection context. This, in turn, is stored within the RADIUS context for the specified server, ensuring a persistent secure environment for subsequent communications.

RADIUS packet handling: The format of the RADIUS packet remains consistent with that used in RADIUS over TCP. The application constructs a RADIUS packet using standard methods. Instead of sending it via a TCP socket, the packet is handed over to the TLS layer for secure encapsulation. TLS effectively becomes the transport layer for RADIUS, hence the designation "RADIUS/TLS."

Secure data transmission: The RADIUS packets are securely transmitted over the TLS layer with data encryption/decryption.

This approach ensures that the RADIUS communications are secure, especially in roaming environments where the packets may pass through various administrative domains and untrusted networks. TLS provides a robust security layer, addressing the vulnerabilities associated with traditional RADIUS over UDP.

Optimized RADIUS session control:

- To manage RADIUS sessions effectively, the RADIUS client employs Path MTU discovery before initiating traffic.
- The RADIUS client avoids using the same source socket for both RADIUS/UDP and RADIUS/TLS traffic to different servers.
- Once a TLS session is established, TLS heartbeats monitor connectivity with the server.
- Additionally, an application-layer watchdog algorithm checks server responsiveness. The client proactively closes idle sessions; sessions indicated as inactive by the TLS heartbeat, or those with only watchdog traffic for three timeouts.
- RADIUS sessions are terminated when RADIUS packets fail validation or contain invalid authenticators. When a session fails validation, the session is validated again using the next specified failover mechanism.

Restrictions for RADIUS with TLS Protection

The list provides the restrictions that apply to RADIUS with TLS protection:

- Broadband Network Gateway (BNG) applications are not supported.
- The default destination port for RADIUS over TLS is TCP 2083 for authentication and accounting. There is no support for custom ports.
- The maximum number of concurrent TLS sessions supported is 50.
- RADIUS over TLS supports IPv4 addresses only.
- A combination of TLS, UDP, and DTLS server types under one server group over RADIUS is not recommended.

Supported ciphers

TLS ciphers are encryption algorithms that secure RADIUS traffic. Here are some supported TLS ciphers:

- TLS_AES_256_CGM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_CGM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CGM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CGM_SHA384
- TLS_DHE_RSA_WITH_AES_256_CGM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_EMPTY_RENEGOTIATION_INFO_SCSV

The cipher suite negotiated between the client and the server when both support TLS 1.3 is:

- TLS_AES_256_GCM_SHA384

Configure RADIUS with TLS protection

To configure RADIUS with TLS protection, use the command **radius-server host** with keyword **radsec-server**.

Before you begin

Before configuring RADIUS with TLS protection, complete these steps on the Cisco router. See **Implementing Certification Authority Interoperability** for more information.

1. Configure a trustpoint.
2. Import the CA certificate.
3. Enroll the trustpoint and generate a client certificate on CA.
4. Import the client certificate.

Procedure

Step 1 Enter the hostname or IP address of the RADIUS server.

```
Router(Config)#radius-server host 209.165.201.1 auth-port 2083 acct-port 2083 radsec-server
```

Step 2 Enter the name of the trusted point so that the router can verify certificates issued to peers.

```
Router(config-radius-host)trustpoint test
```

Your router need not enroll with the CA that issued the certificates to the peers.

Step 3 Commit the changes.

```
Router(config-radius-host)#commit
```

Step 4 Verify that TLS is enabled by using the **show radius** command.

```
Router#show radius
Thu Jun 20 11:43:40.863 UTC
```

```
Global dead time: 0 minute(s)
Number of Servers: 3

Server: 209.165.201.1/2083/2083  is UP
  Address family: IPv4
  Total Deadtime: 0s Last Deadtime: 0s
  Timeout: 5 sec, Retransmit limit: 3
  Quarantined: No
  Authentication:
    0 requests, 0 pending, 0 retransmits
    0 accepts, 0 rejects, 0 challenges
    0 timeouts, 0 bad responses, 0 bad authenticators
    0 unknown types, 0 dropped, 0 ms latest rtt
    Throttled: 0 transactions, 0 timeout, 0 failures
    Estimated Throttled Access Transactions: 0
    Maximum Throttled Access Transactions: 0

    Automated TEST Stats:
        0 requests, 0 timeouts, 0 response, 0 pending
  Server-type: TLS
  Accounting:
    0 requests, 0 pending, 0 retransmits
    0 responses, 0 timeouts, 0 bad responses
    0 bad authenticators, 0 unknown types, 0 dropped
    0 ms latest rtt
    Throttled: 0 transactions, 0 timeout, 0 failures
    Estimated Throttled Accounting Transactions: 0
    Maximum Throttled Accounting Transactions: 0

    Automated TEST Stats:
        0 requests, 0 timeouts, 0 response, 0 pending
```

Step 5 Verify the configuration settings by using the **show running-configuration** command.

```
Router#show running-configuration radius-server
Fri Jun 21 02:59:40.238 UTC
radius-server host 209.165.201.1 auth-port 2083 acct-port 2083
radsec-server trustpoint test
!
```

Hold-Down Timer for TACACS+

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Hold-Down Timer for TACACS+	Release 7.4.1	TACACS+ servers provide AAA services to the user. When a TACACS+ server becomes unreachable, the router sends the client request to another server, leading to considerable delay in addressing requests. To prevent this delay, you can set a hold-down timer on the router. The timer gets triggered after the router marks the TACACS+ server as down. During this period, the router does not select the server that is down for processing any client requests. When the timer expires, the router starts using that TACACS+ server for client transactions. This feature improves latency in providing AAA services to the user by limiting the client requests from being sent to unresponsive servers. This feature introduces the **holddown-time** command.

The TACACS+ server is a AAA server with which the router communicates to provide authentication, authorization, and accounting services for users. When a TACACS+ server goes down, the router is not made aware. After sending a AAA request, the client waits for a response from the server for a configured timeout. If the router does not receive a response within that time frame, it sends the request to the next available server or discards the request if no other servers are available. A new request also needs to follow the same procedure in the same order of servers. The overall process results in sending multiple requests to servers that are down and therefore delays the client request from reaching an active server.

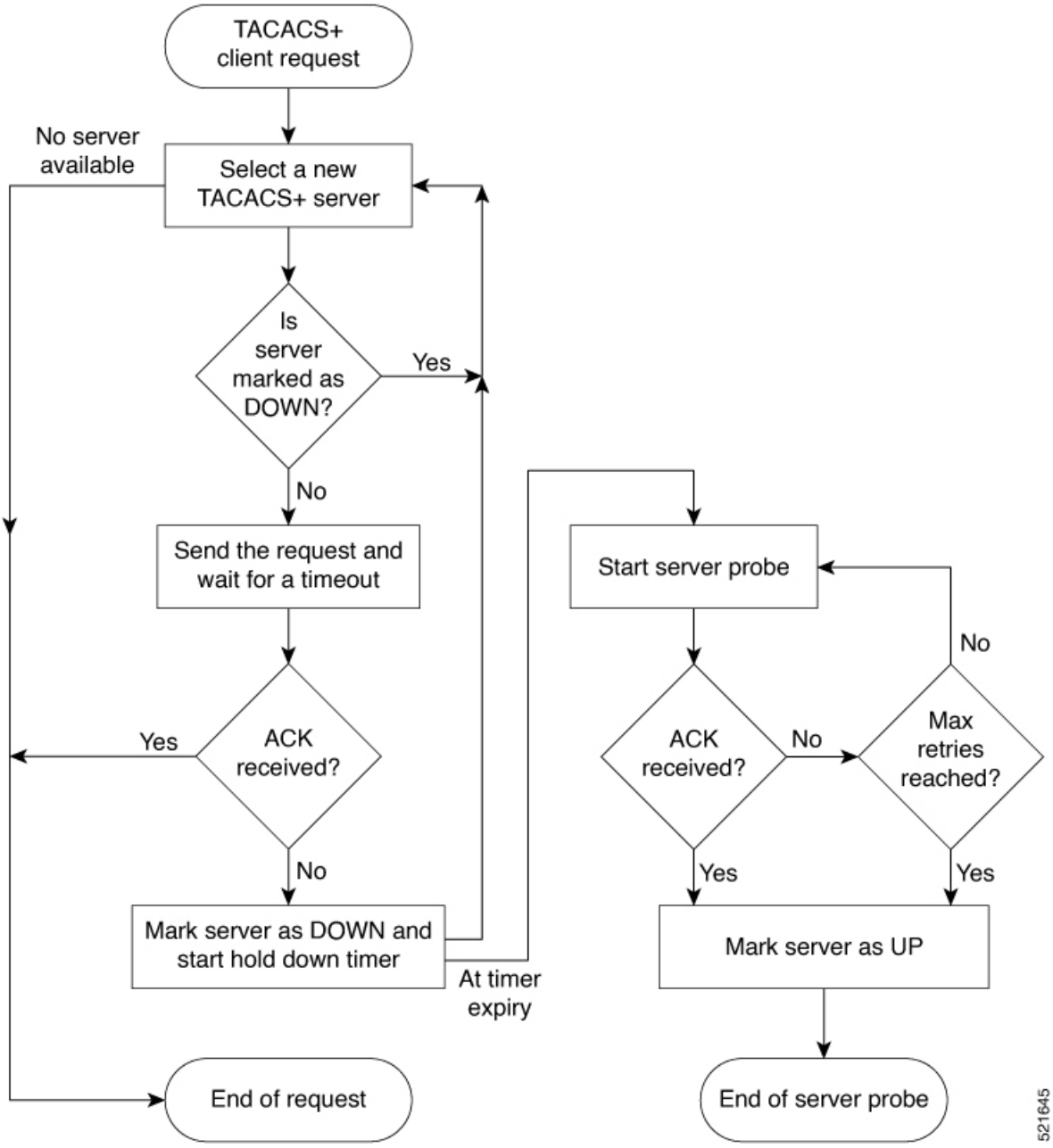
With the TACACS+ hold-down timer feature, you can mark an unresponsive TACACS+ server as down, and also set a duration for which the router does not use that server for further client transaction. After the timer expires, the router starts using that server again for processing client requests. This feature in turn allows you to control the participation of a TACACS+ server in AAA functions, without removing the TACACS+ server configuration from the router.

The hold-down timer value, in seconds, ranges from 0 to 1200. To enable hold-down timer, use the **holddown-time** command under the various configuration modes listed in the How to Configure Hold-Down Timer for TACACS+, on page 58 section.

How Does the Hold-Down Timer for TACACS+ Function?

The following image depicts the functionality of TACACS+ hold-down timer.

Figure 1: Work Flow of TACACS+ Hold-Down Timer



When a TACACS+ client request comes, the router selects a TACACS+ server and checks whether that server is marked as down. If the server is marked as down, the router selects another server until it finds an available server. If the server is not marked as down, the router sends the client request to that server. If the router does not receive an acknowledgment message from the server, it marks that server as down and initiates the hold-down timer. After the timer expires, an internal server probe begins, which checks the connectivity of the down server. The probe tries to connect to the server every 20 seconds, for a maximum of three times (these values are non-configurable). If connection is successful in any of these attempts, then the router marks that server as up, and ends the server probe. Even if the connection fails on all retries of the server probe, the

router still marks the server as up before exiting the server probe. After exiting the server probe, the router considers that server as available again to accept client requests.

If an unresponsive server is still not reachable after the hold-down timer expiry, then the system continues to regard that server as being down, and does not use it for client transactions for some more time (that is, approximately, one minute). The router starts using that server again for further client transactions only after this short delay.

In case the TACACS+ server comes up while the hold-down timer continues, the router continues to consider that server as down until the timer expires.

Model-based AAA

Table 8: Feature History Table

Feature Name	Release Information	Description
NETCONF Access Control Model (NACM) for Protocol Operations and Authorization	Release 7.4.1	NACM is defined in AAA subsystem to manage access control for NETCONF Remote Procedure Calls (RPCs). NACM addresses the need to authenticate the user or user groups, authorize whether the user has the required permission to perform the operation. With this feature, you can configure the authorization rules, groups and rule lists containing multiple groups and rules using CLI commands in addition to existing support for YANG data models. This feature also introduces `Cisco-IOS-XR-um-aaa-nacm-cfg.yang` unified data model to configure user access and privileges. You can access this data model from the Github repository.

The Network Configuration Protocol (NETCONF) protocol does not provide any standard mechanisms to restrict the protocol operations and content that each user is authorized to access. The NETCONF Access Control Model (NACM) is defined in AAA subsystem to manage access-control for NETCONF/YANG RPC requests.

The NACM module provides the ability to control the manageability activities of NETCONF users on the router. You can manage access privileges, the kind of operations that users can perform, and a history of the operations that were performed on the router. The NACM functionality accounts for all the operations that are performed on the box over the NETCONF interface. This functionality authenticates the user or user groups and authorizes permissions for users to perform the operation.

Prerequisites for Model Based AAA

Working with the model based AAA feature requires prior understanding of the following :

- NETCONF-YANG
- RFC 6536: Network Configuration Protocol (NETCONF) Access Control Model

Initial Operation

These are the NACM default values. By default a user is denied write permission, hence you'll not be able to edit the NACM configurations after enabling NACM authorization using AAA command.

```
<enable-nacm>false</enable-nacm>
<read-default>permit</read-default>
<write-default>deny</write-default>
<exec-default>permit</exec-default>
<enable-external-groups>true</enable-external-groups>
```

Therefore we recommend to enable NACM after configuring the required NACM configurations, or after changing the default NACM configurations. Here are few sample configurations:



Note If `access-denied` message is returned while writing NACM configurations, then NACM authorization can be disabled to edit the NACM configurations.

```
<aaa xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-lib-cfg">
<usernames xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-cfg">
<username>
<ordering-index>3</ordering-index>
<name>username</name>
<password>password</password>
     <usergroup-under-usernames>
      <usergroup-under-username>
       <name>root-lr</name>
      </usergroup-under-username>
      <usergroup-under-username>
       <name>cisco-support</name>
      </usergroup-under-username>
     </usergroup-under-usernames>
</username>
</usernames>
</aaa>

<nacm xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-nacm-cfg">
<read-default>permit</read-default>
<write-default>permit</write-default>
<exec-default>permit</exec-default>
<enable-external-groups>true</enable-external-groups>
<groups>
    <group>
    <name>nacm_group</name>
    <user-name>lab</user-name>
    </group>
</groups>
<rule-list>
<name>Rule-list-1</name>
<group>Group_nacm_0_test</group>
<rule>
    <name>Rule-1</name>
    <access-operations>read</access-operations>
    <action>permit</action>
    <module-name>ietf-netconf-acm</module-name>
    <rpc-name>edit-config</rpc-name>
```

```
            <access-operations>*</access-operations>
            <path>/</path>
            <action>permit</action>
        </rule>
    </rule-list>
</nacm>
```

The NACM configuration allows to choose the precedence of external groups over the local groups.

NACM Configuration Management and Persistence

The NACM configuration can be modified using NETCONF or RESTCONF. In order for a user to be able to acccess the NACM configuration, they must have explicit permission to do so, that is, through a NACM rule. Configuration under the /nacm subtree persists when the **copy running-config startup-config** EXEC command is issued, or the **cisco-ia:save-config** RPC is issued.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<save-config xmlns="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

Overview of Configuring NACM

Here are the steps involved in configuring NACM:

1. Configure all NACM rules
2. Enable NACM
3. Disconnect all active NETCONF sessions
4. Launch new NETCONF session



Note Enabling or disabling NACM does not affect any existing NETCONF sessions.

NACM Rules

As per the RFC 6536, NACM defines two categories of rules:

- Global Rules—It includes the following:
  - Enable/Disable NACM
  - Read-Default
  - Write-Default
  - Exec-Default
  - Enable External Groups
- Access Control Rules—It includes the following:
  - Module (used along with protocol rule / data node rule)
  - Protocol
  - Data Node

The following table lists the rules and access operations:

Operation	Description
all	Rule is applied to all types of protocol operations
create	Rule is applied to all protocol operations, which create a new data node such as edit-config operation
read	Rule is applied to all protocol operations, which reads the data node such as get, get-config or notification
update	Rule is applied to all protocol operations, which alters a data node such as edit-config operation
exec	Rule is applied to all exec protocol access operations such as action RPC
delete	Rule is applied to all protocol operations that removes a data node

Note Before enabling NACM using NETCONF RPC, any user with access to the system can create NACM groups and rules. However, after NACM is enabled, only authorised users can change the NACM configurations.

Note Only users who belong to `root-lr` group or with write access in `aaa task` group can enable or disable NACM using CLI commands.

Example: Configure Global Rules

YANG Data Model: You must configure NACM groups and NACM rulelist before configuring NACM rules. The following sample configuration shows a NACM group configuration:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" >
<edit-config>
  <target><candidate/></target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <groups>
      <group>
        <name>group1</name>
        <user-name>user1</user-name>
        <user-name>user2</user-name>
        <user-name>user3</user-name>
      </group>
    </groups>
  </nacm>
</config>
</edit-config>
</rpc>
```

The following sample configuration shows a NACM rule list configuration:

```
<rpc
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"message-id="101">
<edit-config>
  <target>
    <candidate/>
  </target>
 <config>
  <nacm xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-nacm-cfg">
  <rulelist-classes>
  <rulelist-class>
  <ordering-index>1</ordering-index>
 <rulelist-name>GlobalRule</rulelist-name>
 <group-names>
  <group-name>root-system</group-name>
   <group-name>AdminUser</group-name>
 </group-names>
 </rulelist-class>
 </rulelist-classes>
</nacm>
</config>
</edit-config>
</rpc>
```

You can configure the NACM rule list using CLI commands in addition to configuring using YANG data models. The following commands are supported:

```
Router(config)#nacm rule-list 1 GlobalRule
Router(config-rlst)#groupnames root-system AdminUser
```

Example: Configure NACM Global Rules

YANG Data Model:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" >
<edit-config>
  <target><candidate/></target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <read-default>permit</read-default>
    <write-default>permit</write-default>
    <exec-default>permit</exec-default>
    <enable-external-groups>false</enable-external-groups>
  </nacm>
</config>
</edit-config>
</rpc>
```

CLI Command: You can configure the NACM global rules using CLI commands in addition to configuring using YANG data models. The following commands are supported:

```
Router(config)#nacm read-default [ permit | deny ]
Router(config)#nacm write-default [ permit | deny ]
Router(config)#nacm exec-default [ permit | deny ]
Router(config)#nacm enable-external-groups [ true | false ]
```

Note You must have NACM task permissions to make changes.

Example: Configure Access Control Rules

YANG Data Model:

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" >
<edit-config>
<target><candidate/></target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <rule-list>
     <name>GlobalRule</name>
     <rule>
       <name>rule1</name>
       <module-name>ietf-netconf-acm</module-name>
       <rpc-name>edit-config</rpc-name>
       <access-operations>*</access-operations>
       <action>permit</action>
     </rule>
    <rule>
      <name>rule2</name>
      <module-name>ietf-netconf-acm</module-name>
      <rpc-name>get-config</rpc-name>
      <access-operations>create read update exec</accessoperations>
      <action>permit</action>
    </rule>
   </rule-list>
  </nacm>
 </config>
 </edit-config>
</rpc>
```

Note '*' refers to all operations.

CLI Command: You can onfigure the NACM protocol rules using CLI commands in addition to configuring using YANG data models:

```
Router(config)#nacm rule-list 1 GlobalRule
Router(nacm-rlst)#groupnames AdminUser
Router(nacm-rlst)#rule 1 rule1
Router(nacm-rule)#action permit
Router(nacm-rule)#module-name ietf-netconf-acm
Router(nacm-rule)#rule-type rpc edit-config
Router(nacm-rule)#access-operations create read update exec
Router(nacm-rlst)#rule 2 rule2
Router(nacm-rule)#action deny
Router(nacm-rule)#module-name ietf-netconf-acm
Router(nacm-rule)#rule-type rpc get-config
Router(nacm-rule)#access-operations create read update exec
```

Example: Configure NACM Data Node Rules

```
<rpc message-id="101"xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" >
<edit-config>
<target><candidate/></target>
  <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
     <rule-list>
        <name>GlobalRule</name>
```

```
      <rule>
       <name>rule4</name>
       <module-name>*</module-name>
       <path>/nacm/groups/group</path>
       <access-operations>*</access-operations>
       <action>permit</action>
      </rule>
      <rule>
       <name>rule5</name>
       <module-name>ietf-netconf-acm</module-name>
       <path>/nacm/rule-list</path>
       <access-operations>read</access-operations>
       <action>deny</action>
      </rule>
     </rule-list>
   </nacm>
  </config>
 </edit-config>
</rpc>
```



Note '*' refers to all modules, and all operations.

CLI Command: You can configure the NACM data rules using CLI commands in addition to configuring using YANG data models. The following commands are supported:

```
nacm rule-list 1 GlobalRule
 groupnames AdminUser
  rule 4 rule4
   action permit
   module-name *
   rule-type data-node /nacm/groups/group
   access-operations all
  rule 5 rule5
   action deny
   module-name ietf-netconf-acm
   rule-type data-node /nacm/rule-list
   access-operations all
```

Enabling NACM

NACM is disabled on the router by default. Users with root-lr or 'aaa' write task privilege users can enable/disable the NACM via CLI.

To enable NACM, use the following command in the Global configuration mode:

```
Router(config)#aaa authorization nacm default local
```

Cisco IOS XR Software Release 7.4.1 introduces support for external group names.

The external group names are added to the list of local group names to determine the access control rules. External group names are preferred from the list:

```
Router(config)#aaa authorization nacm default prefer-external group tacacs+ local
```

The `local` keyword refers to the locald (AAA local database) and not the NACM database.

Only external group names will be used to determine the access control rules:

```
Router(config)#aaa authorization nacm default only-external local
```

Verification

Use the **show nacm summary** command to verify the default values after enabling NACM:

```
Router# show nacm summary
Mon Jan 15 16:47:43.549 UTC
 NACM SUMMARY
 ------------------------------------------------------------------------------
 Enable Nacm : True
 Enable External Groups : True
 Number of Groups : 0
 Number of Users : 0
 Number of Rules : 0
 Number of Rulelist : 0
 Default Read : permit
 Default Write : deny
 Default Exec : permit
 Denied Operations : 0
 Denied Data Writes : 0
 Denied Notifications : 0
```

Associated Commands

- Router#**show nacm summary**
- Router#**show nacm users [user-name]**
- Router#**show nacm rule-list [rule-list-name] [rule [rule-name]]**
- Router#**show nacm groups [group-name]secret**

Verify the NACM Configurations

Use the **show nacm summary** command to verify the NACM configurations:

```
Router# show nacm summary
Mon Jan 15 17:02:46.696 UTC
 NACM SUMMARY
 ------------------------------------------------------------------------------
 Enable Nacm : True
 Enable External Groups : True
 Number of Groups : 3
 Number of Users : 3
 Number of Rules : 4
 Number of Rulelist : 2
 Default Read : permit
 Default Write : permit
 Default Exec : permit
 Denied Operations : 1
 Denied Data Writes : 0
 Denied Notifications : 0
------------------------------------------------------------------------------
```

Associated Commands

- Router#**show nacm summary**
- Router#**show nacm users [user-name]**
- Router#**show nacm rule-list [rule-list-name] [rule [rule-name]]**

- Router#**show nacm groups [group-name]secret**

Disabling NACM

There are two ways you can disable NACM. Use one of the following commands:

Configuring NACM authorization as none:

```
Router(config)# aaa authorization nacm default none
```

or

Using no form of AAA authorization command:

```
Router(config)# no aaa authorization nacm default
```

Verification

Use the **show nacm summary** command to verify the default values after disabling NACM:

```
Router# show nacm summary

Mon Jan 15 17:02:46.696 UTC
 NACM SUMMARY
 -------------------------------------------------------------------------------
 Enable Nacm : False
 Enable External Groups : True
 Number of Groups : 0
 Number of Users : 0
 Number of Rules : 0
 Number of Rulelist : 0
 Default Read : permit
 Default Write : deny
 Default Exec : permit
 Denied Operations : 0
 Denied Data Writes : 0
 Denied Notifications : 0
```

Command Authorization Using Local User Account

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Command Authorization Using Local User Account	Release 7.5.1	This feature allows locally authenticated users—authenticated by the AAA server internal to the router—to run all XR VM commands even if a remote TACACS+ AAA server is not reachable for authorization. It prevents a complete router lockdown. The feature also prevents remotely authenticated users—authenticated using a remote AAA server (say, TACACS+ server)—from running any non-permitted commands on the router, and thus prevents misuse of user privileges. This feature modifies the **aaa authorization commands default** command to include the **local** option for XR VM command authorization.

Currently, when a user tries to execute a command on XR VM, the router checks to see whether the user has required permissions to execute it. The router does this authorization process in two steps. First, the system compares the task-IDs of the user with the required task-IDs for the command. If the user has all required task-IDs, and if AAA authorization is configured, then the system sends an authorization request to the local or remote AAA server, based on that configuration. Based on the response from the AAA server, the system allows or rejects the command execution. If authorization is not configured or if it configured with option *none*, then the system bypasses authorization check and allows user to execute the command.

Similarly, the existing remote authorization process using TACACS+ server has two options—remote authorization using *tacacs+* and *none*. The authorization process using *TACACS+* option uses an external TACACS+ server for authorization. The authorization using *none* option allows the user to execute the command without any authorization check. TACACS+ authorization has the advantage of fine-tuning authorization rules and providing more control on system access that cannot be otherwise done locally. However, if the remote server is not reachable, a user who leverages TACACS+ authorization might get into an unpredictable state of router, as mentioned in these scenarios:

- Remote authorization using *TACACS+* with failover option as *none* (that is, with the **aaa authorization commands default group tacacs+ none** configuration)

  If TACACS+ server is not reachable, then the system bypasses the authorization check and allows user to execute the command. A user who does not have permission to execute certain commands due to additional authorization rules on the TACACS+ server, then gets permission to execute those commands in this scenario. This action introduces a privilege escalation.

- Remote authorization using *TACACS+* without any failover option (that is, with the **aaa authorization commands default group tacacs+** configuration)

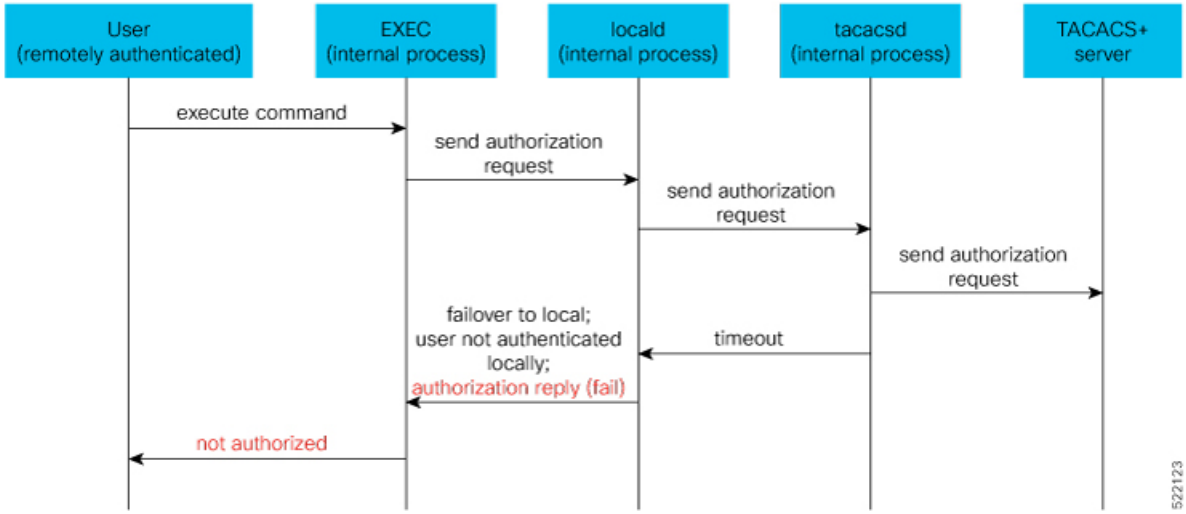
  If TACACS+ server is not reachable, then the system does not authorize the command at all. Because the user then cannot execute any command, the router gets locked out.

With the introduction of command authorization using local user account feature in Cisco IOS XR Software Release 7.5.1, locally authenticated users can execute commands even if a TACACS+ server is not reachable. This behavior is similar to the behavior with the failover option *none*, with the only difference that only locally authenticated users can execute commands in this case. This functionality thereby prevents a complete lockdown of the router as mentioned in one of the previously existing scenarios mentioned earlier. At the same time, the feature also prevents users who are authenticated remotely (that is, TACACS+ authenticated users) from executing any non-permitted command on the router. This behavior in turn helps to prevent any sort of misuse of user privileges on the router.

Call Flow of Command Authorization

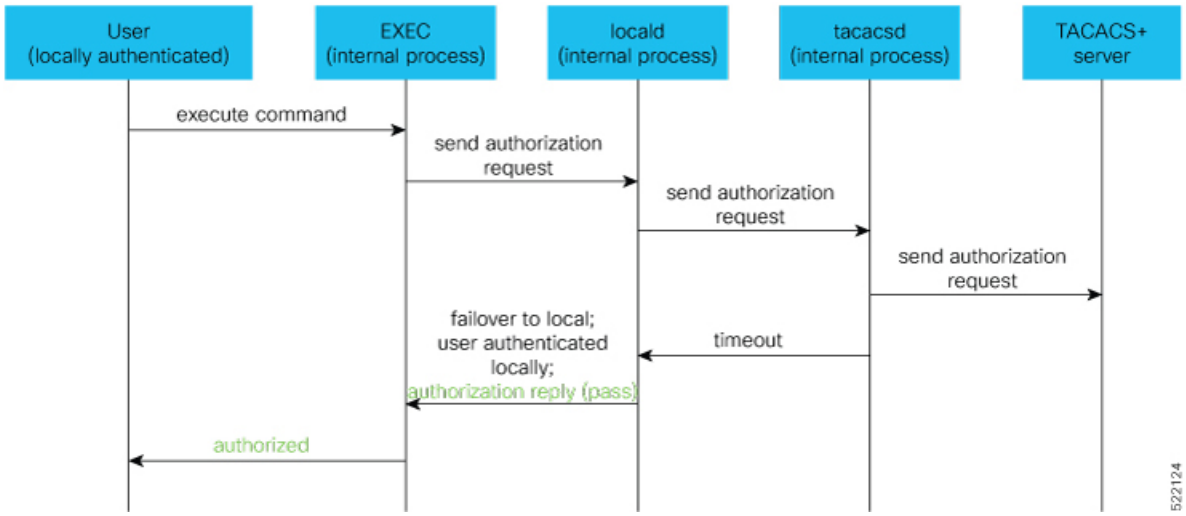
Consider a scenario where the user is remotely authenticated. In the event of timeout from the TACACS+ server, the command authorization fails. The user cannot execute any command until the TACACS+ server is reachable again, thereby preventing misuse of user privileges on the router.

Figure 2: Call Flow of Command Authorization for Remotely Authenticated Users



Consider a scenario where the user is locally authenticated. The command authorization still succeeds even if the authorization request to the TACACS+ server times out. There is no additional check done by the local AAA component in the router. As a result, the user can execute the command irrespective of the fact that the TACACS+ server is not reachable. This functionality prevents a complete lockdown of the router.

Figure 3: Call Flow of Command Authorization for Locally Authenticated Users



Configure Command Authorization Using Local User Account

Guidelines

Although there is no restriction in configuring local command authorization, you must be cautious to prevent any potential lockout due to misconfiguration. For instance, if *local* is the only method of authorization specified for the commands, a remotely authenticated user configuring command authorization using local user account feature cannot execute further commands.

Configuration Example

To configure command authorization using local user account, use the **local** option in the **aaa authorization** command in any of these formats:

```
Router#configure
Router(config)#aaa authorization commands default group tacacs+ local
```

Or

```
Router(config)#aaa authorization commands default local
```

Running Configuration

```
Router#show run aaa
!
aaa authorization commands default group tacacs+ local
!
```

```
Router#show run aaa
!
aaa authorization commands default local
!
```

Verification

```
Router#show user authentication method
local
```

Feature Behavior and Use Case Scenarios

Feature Behavior With Various Local Command Authorization Options

This table lists the feature behavior scenarios with various local command authorization options.

Table 10: Feature Behavior with Various Local Command Authorization Options

AAA Configuration	Expected Behavior
aaa authorization commands default group tacacs+ local	If TACACS+ server is not reachable, system allows locally authenticated users to execute the command. If TACACS+ server is reachable and if it returns an authorization failure, then the system does not perform any failover to local authentication with this configuration.
aaa authorization commands default local	This configuration allows only locally authenticated users to execute commands. System completely blocks remote users from executing any command.
aaa authorization commands default local group tacacs+	In this scenario, system chooses local authorization first and grants access if the user is locally authenticated. If not, the request fails over to TACACS+ server. This combination of command options is useful when both local and remote authenticated users want to execute commands when TACACS+ server is reachable.
aaa authorization commands default local none	Although configurable, this combination of command options does not provide any additional security with respect to user access. It is equivalent to having no authorization.

Use Case Scenarios of Command Authorization

In the following scenarios, local user refers to user whose is authenticated locally and whose profile is available locally, but not available on the remote server (TACACS+ server). Similarly, remote user refers to user whose is authenticated remotely and whose profile is available on the remote server, but not available locally. And, both local user and remote user are considered to have *root-lr* permission to execute the commands, in these scenarios.

Table 11: Use Case Scenarios of Command Authorization

Type of User (local or remote)	AAA Configuration Summary	Use Case Scenario	Expected Behavior
Local and remote user	No command authorization configured	Execute a command	Command authorization succeeds if the required task-IDs are available
Local user	Only *tacacs+ command authorization* configured.	Execute a command when TACACS+ server is reachable	Command authorization fails
		Execute a command when TACACS+ server is not reachable	Command authorization fails

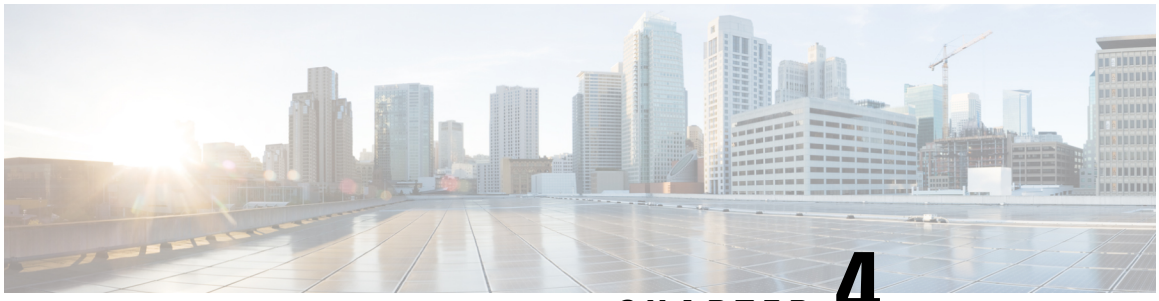
Remote user	Only *tacacs+ command authorization* configured	Execute a command when TACACS+ server is reachable	Command authorization succeeds
			`Router#show run aaa authorization`
			`aaa authorization commands default group tacacs+`

		Execute a command when TACACS+ server is not reachable	Command authorization fails
Local user	Only *tacacs+ command authorization* configured with failover option as *none*.	Execute a command when TACACS+ server is reachable	Command authorization fails

		Execute a command when TACACS+ server is not reachable	Command authorization succeeds
			`Router#show user authentication method`
			`local`

Remote user	Only *tacacs+ command authorization* configured with failover option as *none*.	Execute a command that is restricted only to that user when TACACS+ server is reachable	Command authorization fails
		Execute a command that is restricted only to that user when TACACS+ server is not reachable	Command authorization succeeds

Type of User (local or remote)	AAA Configuration Summary	Use Case Scenario	Expected Behavior
Local user	Only *local command authorization* configured.	Execute a command	Command authorization succeeds `Router#`**`show run aaa authentication`** `aaa authentication login default group tacacs+ local`
Remote user	Only *local command authorization* configured.	Execute a command	Command authorization fails
Local user	Only *tacacs+ command authorization* configured with failover option as *local*.	Execute a command when TACACS+ server is reachable	Command authorization fails
		Execute a command when TACACS+ server is not reachable	Command authorization succeeds `Router#`**`show run aaa authorization`** `aaa authorization commands default group tacacs+ local`
Remote user	Only *tacacs+ command authorization* configured with failover option as *local*.	Execute a command when TACACS+ server is reachable	Command authorization succeeds `Router#`**`show run aaa authorization`** `aaa authorization commands default group tacacs+ local`
		Execute a command when TACACS+ server is not reachable	Command authorization fails



CHAPTER 4

Implementing Certification Authority Interoperability

CA interoperability permits devices and CAs to communicate so that your device can obtain and use digital certificates from the CA. Although IPSec can be implemented in your network without the use of a CA, using a CA provides manageability and scalability for IPSec.

Note IPSec will be supported in a future release.

• Implementing Certification Authority Interoperability, on page 103

Implementing Certification Authority Interoperability

CA interoperability permits devices and CAs to communicate so that your device can obtain and use digital certificates from the CA. Although IPSec can be implemented in your network without the use of a CA, using a CA provides manageability and scalability for IPSec.

Note IPSec will be supported in a future release.

Prerequisites for Implementing Certification Authority

To successfully implement CA interoperability, these prerequisites must be met:

- **User group assignment**: You must belong to a user group associated with a task group that includes the necessary task IDs for CA-related commands. Refer to the command reference guides for specific task ID requirements. If you encounter issues with command access, contact your AAA administrator for assistance.
- **RPM for security software**: An RPM is a modular component of the software. Instead of one large image, IOS XR is divided into smaller pieces, allowing you to install only what you need. RPMs typically fall into two categories:

- Feature packages: These add specific functionality to the router, such as OSPF, BGP, MPLS, or security features.
- Software Maintenance Upgrades (SMUs): These are patches used to fix specific bugs or security vulnerabilities without requiring a full OS upgrade. In 64-bit XR, SMUs are delivered as RPM files.

- Installation and activation requirements:
  - **Cisco IOS XR software release 7.0.1 and later**: For most platforms, the required functionality is integrated into the base image, eliminating the need for a separate RPM installation.
  - **Specific Product IDs (PIDs)**: Regardless of your Cisco IOS XR software release, RPM installation is mandatory for the following PIDs:
    - N540-28Z4C-SYS-A
    - N540-28Z4C-SYS-D
    - N540X-16Z4G8Q2C-A
    - N540X-16Z4G8Q2C-D
    - N540-12Z20G-SYS-A
    - N540-12Z20G-SYS-D
    - N540X-12Z16G-SYS-A
    - N540X-12Z16G-SYS-D

- **Available Certification Authority (CA)**: A CA must be accessible within your network before you can configure this interoperability feature. This CA must support the Cisco Systems PKI protocol, specifically the Simple Certificate Enrollment Protocol (SCEP), formerly known as CEP.

Restrictions for Implementing Certification Authority

- The software does not support CA server public keys greater than 2048 bits.
- Starting Cisco IOS XR Software Release 7.4.1, we mandate the below X509 certificate Subject Alternate Name (SAN) fields and domain name server configuration to validate SAN. TLS connection cannot be established if there is no domain name-server is configured.

  Here are some key-points regarding SAN field:
  - SAN must be a fully-qualified domain name. For example, DNS:smartreceiver.cisco.com
  - SAN must be a critical extension in the absence of Common Name (CN).
  - If the SAN cannot be represented as a FQDN, then it must be configured with GeneralName field as IP Address but not as DNS. For example, **IP address: 192.0.2.1**

  To configure domain name-server use the **domain name-server** *ip-address*.

  To configure domain name-server with VRF, use the following commands:
  - **domain vrf name-server***ip-address*

- Use the **crypto ca** *trustpoint-name* **vrf** *vrf-name* command when you are using VRF.
- Use the **crypto ca trustpoint Trustpool vrf** *vrf-name* command for smart-licensing.

For Static Domain Name Configuration, use the **domain ipv4 host** *host-name ip-address* command. and for configuring static domain name using VRF, use the **domain ipv4 vrf** *vrf-name host-name ip-address* command.

- Starting Cisco IOS XR Software Release 7.4.2, you can bypass FQDN and IP address check in SAN by configuring **crypto ca fqdn-check ip-address allow.**
- Starting Cisco IOS XR Software Release 7.10.1, in addition to SAN field, you must also set FQDN in the CN field. However, CN check can be bypassed by configuring **crypto ca fqdn ip-address allow.** For example, CN=root.cisco.com.

  To ensure successful FQDN validations for **SAN & CN** fields, the device must have proper DNS configurations in place. If the DNS server becomes unreachable, it's mandatory to configure a static DNS entry. Failure to do so will result in certificate validation errors, leading to potential TLS authentication failures.



Note Cisco strongly recommends regenerating the existing certificates with valid FQDNs.

Configure Router Hostname and IP Domain Name

This task configures a router hostname and IP domain name.

You must configure the hostname and IP domain name of the router if they have not already been configured. The hostname and IP domain name are required because the router assigns a fully qualified domain name (FQDN) to the keys and certificates used by IPSec, and the FQDN is based on the hostname and IP domain name you assign to the router. For example, a certificate named router20.example.com is based on a router hostname of router20 and a router IP domain name of example.com.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **hostname** *name*

Example:

```
RP/0/RP0/CPU0:router(config)# hostname myhost
```

Configures the hostname of the router.

Step 3 **domain name domain-name**

Example:

```
RP/0/RP0/CPU0:router(config)# domain name mydomain.com
```

Configures the IP domain name of the router.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

RSA key pairs

RSA key pairs play a crucial role in securing IKE key management messages by enabling both signing and encryption. Their use is required before a certificate can be obtained for your router. Specifically, RSA key pairs are used to

- digitally sign IKE key management messages, ensuring their authenticity and integrity
- encrypt IKE key management messages, protecting sensitive information from unauthorized access, and
- satisfy prerequisites for obtaining a digital certificate for the router, which further enhances security in network communications.

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
Syslog warnings for RSA keys and DSA keys	Release 26.1.1	This enhancement ensures Cisco IOS XR devices remain compliant with evolving security standards. During system reboot, the device now sends syslog warnings if weak SSH host keys are detected—specifically, RSA keys less than 3072 bits or any DSA keys. Additionally, the default RSA key size has been increased from 2048 to 3072 bits to further strengthen security. This feature introduces these changes: CLI: • The **crypto key generate rsa** command has been modified. • The **crypto key generate dsa** command has been modified. • The **show crypto key mypubkey rsa** command has been modified.

Syslog warnings for RSA keys and DSA keys

Starting with Cisco IOS XR Release 26.1.1, the default RSA key size is 3072 bits; syslog warning is triggered during system boot for weak SSH host key, where RSA key is less than 3072 bits. Additionally, DSA keys are no longer auto-generated at boot, and if found, a syslog warning prompts their removal. These changes ensure compliance with current cryptographic security standards and enhance device protection against emerging threats.

Generate RSA Key Pair

This task generates an RSA key pair.

From Cisco IOS XR Software Release 7.0.1 and later, the crypto keys are auto-generated at the time of router boot up. Hence, step 1 is required to be configured only if the RSA host-key pair is not present in the router under some scenarios.

RSA key pairs are used to sign and encrypt IKE key management messages and are required before you can obtain a certificate for your router.

Procedure

Step 1 Execute the **crypto key generate rsa general-keys** command to generate general purpose RSA key pairs.

Example:

```
RP/0/RP0/CPU0:router# crypto key generate rsa general-keys
```

From Cisco IOS XR Release 7.3.2 onwards, you can configure this command from XR Config mode.

Step 2 Execute the **crypto key zeroize rsa** [*keypair-label*] to delete all RSAs from the router.

Example:

```
RP/0/RP0/CPU0:router# crypto key zeroize rsa key1
```

- Under certain circumstances, you may want to delete all RSA keys from you router. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys.
- To remove a specific RSA key pair, use the *keypair-label* argument.

Step 3 Execute the **show crypto key mypubkey rsa** command to display the RSA public keys for your router.

Example:

```
Router# show crypto key mypubkey rsa
Fri Mar 27 14:00:20.954 IST
Key label: the_default
Type : RSA General purpose
Size : 3072
Created : 01:13:10 IST Thu Feb 06 2025
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
 B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
 F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
 39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
 AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
 9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
 EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
 22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
 11020301 0001
```

Import Public Key to the Router

This task imports a public key to the router.

A public key is imported to the router to authenticate the user.

Procedure

Step 1 **crypto key import authentication rsa** [**usage keys** | **general-keys**] [**keypair-label**]

Example:

```
RP/0/RP0/CPU0:router# crypto key import authentication rsa general-keys
```

Generates RSA key pairs.

- Use the **usage keys** keyword to specify special usage keys; use the **general-keys** keyword to specify general- purpose RSA keys.
- The *keypair-label* argument is the RSA key pair label that names the RSA key pairs.

Step 2 show crypto key mypubkey rsa

Example:

```
RP/0/RP0/CPU0:router# show crypto key mypubkey rsa
```

(Optional) Displays the RSA public keys for your router.

Declare Certification Authority and Configure Trusted Point

This task declares a CA and configures a trusted point.

Warning If a `commit replace` operation removes a crypto CA trustpoint or certificate configuration, the system can clear the installed certificates when it processes the delete event. If the operation later fails and the configuration is restored, the certificates might not be restored. Services that use the certificates can become unavailable. After a failed `commit replace`, verify that the required certificates are still installed with **show crypto ca certificates** comamnd and reinstall any certificates that were cleared.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **crypto ca trustpoint ca-name**

Example:

```
Router(config)# crypto ca trustpoint myca
```

Declares a CA.

- Configures a trusted point with a selected name so that your router can verify certificates issued to peers.
- Enters trustpoint configuration mode.

Note
If you want to do certificate enrolment when the server or destination is in a VRF, use the following command after step 2 to configure the VRF:

```
Router(config-trustp)# vrf vrf-name
```

Step 3 **enrollment url CA-URL**

Example:

```
Router(config-trustp)# enrollment url http://ca.domain.com/certsrv/mscep/mscep.dll
```

Specifies the URL of the CA.

- The URL should include any nonstandard cgi-bin script location.

Note
If you want to do certificate enrolment when the destination URL is in a VRF, use the following command instead:

```
Router(config-trustp)# enrollment url tftp-address;vrf-name/ca-name
```

Step 4 **query url LDAP-URL**

Example:

```
Router(config-trustp)# query url ldap://my-ldap.domain.com
```

(Optional) Specifies the location of the LDAP server if your CA system supports the LDAP protocol.

Step 5 **enrollment retry period minutes**

Example:

```
Router(config-trustp)# enrollment retry period 2
```

(Optional) Specifies a retry period.

- After requesting a certificate, the router waits to receive a certificate from the CA. If the router does not receive a certificate within a period of time (the retry period) the router will send another certificate request.
- Range is from 1 to 60 minutes. Default is 1 minute.

Step 6 **enrollment retry count number**

Example:

```
Router(config-trustp)# enrollment retry count 10
```

(Optional) Specifies how many times the router continues to send unsuccessful certificate requests before giving up.

- The range is from 1 to 100.

Step 7 **rsakeypair keypair-label**

Example:

```
Router(config-trustp)# rsakeypair mykey
```

(Optional) Specifies a named RSA key pair generated using the **crypto key generate rsa** command for this trustpoint.

- Not setting this key pair means that the trustpoint uses the default RSA key in the current configuration.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Authenticate CA

This task authenticates the CA to your router.

The router must authenticate the CA by obtaining the self-signed certificate of the CA, which contains the public key of the CA. Because the certificate of the CA is self-signed (the CA signs its own certificate), manually authenticate the public key of the CA by contacting the CA administrator to compare the fingerprint of the CA certificate.

Procedure

Step 1 **crypto ca authenticate ca-name**

Example:

```
RP/0/RP0/CPU0:router# crypto ca authenticate myca
```

Authenticates the CA to your router by obtaining a CA certificate, which contains the public key for the CA.

Step 2 show crypto ca certificates

Example:

```
RP/0/RP0/CPU0:router# show crypto ca certificates
```

(Optional) Displays information about the CA certificate.

Request Your Own Certificates

This task requests certificates from the CA.

You must obtain a signed certificate from the CA for each of your router's RSA key pairs. If you generated general-purpose RSA keys, your router has only one RSA key pair and needs only one certificate. If you previously generated special usage RSA keys, your router has two RSA key pairs and needs two certificates.

Procedure

Step 1 **crypto ca enroll ca-name**

Example:

```
RP/0/RP0/CPU0:router# crypto ca enroll myca
```

Requests certificates for all of your RSA key pairs.

- This command causes your router to request as many certificates as there are RSA key pairs, so you need only perform this command once, even if you have special usage RSA key pairs.
- This command requires you to create a challenge password that is not saved with the configuration. This password is required if your certificate needs to be revoked, so you must remember this password.
- A certificate may be issued immediately or the router sends a certificate request every minute until the enrollment retry period is reached and a timeout occurs. If a timeout occurs, contact your system administrator to get your request approved, and then enter this command again.

Step 2 **show crypto ca certificates**

Example:

```
RP/0/RP0/CPU0:router# show crypto ca certificates
```

(Optional) Displays information about the CA certificate.

Configure Certificate Enrollment Using Cut-and-Paste

This task declares the trustpoint certification authority (CA) that your router should use and configures that trustpoint CA for manual enrollment by using cut-and-paste.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **crypto ca trustpoint** *ca-name*

Example:

```
 RP/0/RP0/CPU0:router(config)# crypto ca trustpoint myca RP/0//CPU0:router(config-trustp)#
```

Declares the CA that your router should use and enters trustpoint configuration mode.

- Use the *ca-name* argument to specify the name of the CA.

Step 3 enrollment terminal

Example:

```
RP/0/RP0/CPU0:router(config-trustp)# enrollment terminal
```

Specifies manual cut-and-paste certificate enrollment.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5 **crypto ca authenticate** *ca-name*

Example:

```
RP/0/RP0/CPU0:router# crypto ca authenticate myca
```

Authenticates the CA by obtaining the certificate of the CA.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in step 2.

Step 6 **crypto ca enroll** *ca-name*

Example:

```
RP/0/RP0/CPU0:router# crypto ca enroll myca
```

Obtains the certificates for your router from the CA.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2.

Step 7 **crypto ca import** ca- *name* **certificate**

Example:

```
RP/0/RP0/CPU0:router# crypto ca import myca certificate
```

Imports a certificate manually at the terminal.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2.

Note
You must enter the **crypto ca import** command twice if usage keys (signature and encryption keys) are used. The first time the command is entered, one of the certificates is pasted into the router; the second time the command is entered, the other certificate is pasted into the router. (It does not matter which certificate is pasted first.

Step 8 show crypto ca certificates

Example:

```
RP/0/RP0/CPU0:router# show crypto ca certificates
```

Displays information about your certificate and the CA certificate.

The following example shows how to configure CA interoperability.

Comments are included within the configuration to explain various commands.

```
configure
hostname myrouter
domain name mydomain.com
end

Uncommitted changes found, commit them? [yes]:yes

crypto key generate rsa mykey

The name for the keys will be:mykey
Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose
Keypair
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus [1024]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]

show crypto key mypubkey rsa

Key label:mykey
Type     :RSA General purpose
Size     :1024
Created  :17:33:23 UTC Thu Sep 18 2003
Data     :
 30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00CB8D86
 BF6707AA FD7E4F08 A1F70080 B9E6016B 8128004C B477817B BCF35106 BC60B06E
 07A417FD 7979D262 B35465A6 1D3B70D1 36ACAFBD 7F91D5A0 CFB0EE91 B9D52C69
 7CAF89ED F66A6A58 89EEF776 A03916CB 3663FB17 B7DBEBF8 1C54AF7F 293F3004
 C15B08A8 C6965F1E 289DD724 BD40AF59 E90E44D5 7D590000 5C4BEA9D B5020301
 0001

! The following commands declare a CA and configure a trusted point.

configure
crypto ca trustpoint myca
enrollment url http://xyz-ultra5
enrollment retry count 25
enrollment retry period 2
rsakeypair mykey
end

Uncommitted changes found, commit them? [yes]:yes

! The following command authenticates the CA to your router.

crypto ca authenticate myca

Serial Number  :01
Subject Name   :
```

```
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
Issued By      :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
Validity Start :07:00:00 UTC Tue Aug 19 2003
Validity End   :07:00:00 UTC Wed Aug 19 2020
Fingerprint:58 71 FB 94 55 65 D4 64 38 91 2B 00 61 E9 F8 05
Do you accept this certificate?? [yes/no]:yes

! The following command requests certificates for all of your RSA key pairs.

crypto ca enroll myca

% Start certificate enrollment ...
% Create a challenge password. You will need to verbally provide this
  password to the CA Administrator in order to revoke your certificate.
% For security reasons your password will not be saved in the configuration.
% Please make a note of it.

Password:
Re-enter Password:
    Fingerprint: 17D8B38D ED2BDF2E DF8ADBF7 A7DBE35A

! The following command displays information about your certificate and the CA certificate.

show crypto ca certificates

Trustpoint       :myca
============================================================
CA certificate
  Serial Number  :01
  Subject Name   :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Validity Start :07:00:00 UTC Tue Aug 19 2003
  Validity End   :07:00:00 UTC Wed Aug 19 2020
Router certificate
  Key usage      :General Purpose
  Status         :Available
  Serial Number  :6E
  Subject Name   :
        unstructuredName=myrouter.mydomain.com,o=Cisco Systems
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Validity Start :21:43:14 UTC Mon Sep 22 2003
  Validity End   :21:43:14 UTC Mon Sep 29 2003
  CRL Distribution Point
        ldap://coax-u10.cisco.com/CN=Root coax-u10 Certificate Manager,O=Cisco Systems
```

Certificate Authority Trust Pool Management

The trust pool feature is used to authenticate sessions, such as HTTPS, that occur between devices by using commonly recognized trusted agents called certificate authorities (CAs). This feature is enabled by default in the software to create a scheme to provision, store, and manage a pool of certificates from known CAs in a way similar to the services a browser provides for securing sessions. A special trusted point called a trust pool is designated, containing multiple known CA certificates from Cisco and possibly from other vendors. The trust pool consists of both built-in and downloaded CA certificates.

Implementing Certification Authority Interoperability provides details on Certificate Authority and trusted point.

CA Certificate Bundling in the Trust Pool

The router uses a built-in CA certificate bundle that is packaged into the asr9k-k9sec PIE. The bundle is contained in a special certificate store called a CA trust pool, which is updated automatically by Cisco. This trust pool is known by Cisco and other vendors. A CA certificate bundle can be in the following formats:

- Privilege Management Infrastructure (PMI) certificates in Distinguished Encoding Rules (DER) binary format enveloped within a public-key cryptographic message syntax standard 7 (pkcs7).
- A file containing concatenated X.509 certificates in Privacy Enhanced Mail (PEM) format with PEM headers.

Updating the CA Trustpool

The CA trustpool must be updated when the following conditions occur:

- A certificate in the trustpool is due to expire or has been reissued.
- The published CA certificate bundle contains additional trusted certificates that are needed by a given application.
- The configuration has been corrupted.

The CA trustpool is considered as a single entity, As such, any update you perform will replace the entire trustpool.

Note A built-in certificate in the trustpool cannot be physically replaced. However, a built-in certificate is rendered inactive after an update if its X.509 subject-name attribute matches the certificate in the CA certificate bundle.

Following are the methods available for updating the certificates in the trustpool:

- **Automatic update:** A timer is established for the trustpool that matches the CA certificate with the earliest expiration time. If the timer is running and a bundle location is not configured and not explicitly disabled, syslog warnings should be issued at reasonable intervals to alert the admin that this trustpool policy option is not set. Automatic trustpool updates use the configured URL. When the CA trustpool expires, the policy is read, the bundle is loaded, and the PKI trustpool is replaced. If the automatic CA trustpool update encounters problems when initiating, then the following schedule is used to initiate the update until the download is successful: 20 days, 15 days, 10 days, 5 days, 4 days, 3 days, 2 days, 1 day, and then once every hour.
- **Manual update:** Manually Update Certificates in Trust Pool, on page 116 provides details.

Manually Update Certificates in Trust Pool

The CA trust pool feature is enabled by default and uses the built-in CA certificate bundle in the trust pool, which receives automatic updates from Cisco. Perform this task to manually update certificates in the trust pool if they are not current, are corrupt, or if certain certificates need to be updated.

Procedure

	Command or Action	Purpose
Step 1	**crypto ca trustpool import url clean** **Example:** `RP/0/RP0/CPU0:IMC0#crypto ca trustpool import url clean`	(Optional) Manually removes all downloaded CA certificates. This command is run in the EXEC mode.
Step 2	**crypto ca trustpool import url `url`** **Example:** `RP/0/RP0/CPU0:IMC0#crypto ca trustpool import url http://www.cisco.com/security/pki/trs/ios.p7b`	Specify the URL from which the CA trust pool certificate bundle must be downloaded. This manually imports (downloads) the CA certificate bundle into the CA trust pool to update or replace the existing CA certificate bundle.

Step 3	**show crypto ca trustpool policy**
	Example:
	`RP/0/RP0/CPU0:IMC0#show crypto ca trustpool`
	
	`Trustpool: Built-In`
	`========`
	`CA certificate`
	`  Serial Number  : 5F:F8:7B:28:2B:54:DC:8D:42:A3:15:B5:68:C9:AD:FF`
	
	`  Subject:`
	`  CN=Cisco Root CA 2048,O=Cisco Systems`
	`Issued By  :`
	`  CN=Cisco Root CA 2048,O=Cisco Systems`
	`  Validity Start : 20:17:12 UTC Fri May 14 2004`
	`  Validity End  : 20:25:42 UTC Mon May 14 2029`
	`  SHA1 Fingerprint:`
	
	`DE990CED99E0431F60EDC3937E7CD5BF0ED9E5FA`
	
	`Trustpool: Built-In`
	`========`
	`CA certificate`
	`  Serial Number  : 2E:D2:0E:73:47:D3:33:83:4B:4F:DD:0D:D7:B6:96:7E`
	`Subject:`
	`  CN=Cisco Root CA M1,O=Cisco`
	`  Issued By  :`
	`  CN=Cisco Root CA M1,O=Cisco`
	`  Validity Start : 20:50:24 UTC Tue Nov 18 2008`
	`  Validity End  : 21:59:46 UTC Fri Nov 18 2033`
	`  SHA1 Fingerprint:`
	
	`45AD6BB499011BB4E84E84316A81C27D89EE5CE7`
		Displays the CA trust pool certificates of the router in a verbose format.

Configuring Optional Trustpool Policy Parameters

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

Step 2	**crypto ca trustpool policy**
	Example:
	`RP/0/RP0/CPU0:IMC0(config)#crypto ca trustpool policy`
	`RP/0/RSP0/CPU0:IMC0(config-trustpool)#`
		Enters ca-trustpool configuration mode where commands can be accessed to configure CA trustpool policy parameters.

Step 3	**cabundle url** `URL` **Example:** `RP/0/RP0/CPU0:IMC0(config-trustpool)#cabundle url http://www.cisco.com/security/pki/crl/crca2048.crl`	Specifies the URL from which the CA trustpool certificate bundle is downloaded.
Step 4	**crl optional** **Example:** `RP/0/RP0/CPU0:IMC0(config-trustpool)#crl optional`	Disables revocation checking when the trustpool policy is being used. By default, the router enforces a check of the revocation status of the certificate by querying the certificate revocation list (CRL).
Step 5	**description** `LINE` **Example:** `RP/0/RP0/CPU0:IMC0(config-trustpool)#description Trustpool for Test.`	

Handling of CA Certificates appearing both in Trust Pool and Trust Point

There may be cases where a CA resides in both the trust pool and a trust point; for example, a trust point is using a CA and a CA bundle is downloaded later with this same CA inside. In this scenario, the CA in the trust point and its policy is considered, before the CA in the trust pool or trust pool policy to ensure that any current behavior is not altered when the trust pool feature is implemented on the router.

The policy indicates how the security appliance obtains the CA certificate and the authentication policies for user certificates issued by the CA.

Expiry Notification for PKI Certificate

The section provides information about the notification mechanism using SNMP trap and syslog messages when a public key infrastructure (PKI) certificate is approaching its expiry date.

Learn About the PKI Alert Notification

Security is critical and availability of certificates for applications is vital for authenticating the router. If the certificate expires, they become invalid and impacts services like Crosswork Trust Insights, Internet Key Exchange version 2, dot1x, and so on.

What if there is a mechanism to alert the user about the expiry date of the certificate?

From Release 7.1.1, IOS -XR provides a mechanism by which a CA client sends a notification to a syslog server when certificates are on the verge of expiry. Alert notifications are sent either through the syslog server or Simple Network Management Protocol (SNMP) traps.

PKI traps retrieves the certificate information of the devices in the network. The device sends SNMP traps at regular intervals to the network management system (NMS) based on the threshold configured in the device.

An SNMP trap (certificate expiry notification) is sent to the SNMP server at regular intervals starting from 60 days to one week before the certificate end date. The notifications are sent at the following intervals:

The notifications are sent at the following intervals:

Intervals	Description	Notification Mode
First notification	The notification is sent 60 days before the expiry of the certificate.	The notificiation are in a warning mode.
Repeated notifications	The repeated notification is sent every week, until a week before the expiry of the certificate. The notifications are in a warning mode when the certificate is valid for more than a week.	The notifications are in a warning mode when the certificate is valid for more than a week.
Last notification	The notifications are sent every day until the certificate expiry date.	The notifications are in an alert mode when the validity of a certificate is less than a week.

The notifications include the following information:

- Certificate serial number
- Certificate issuer name
- Trustpoint name
- Certificate type
- Number of days remaining for the certificate to expire
- Certificate subject name

The following is a syslog message that is displayed on the device:

```
%SECURITY-CEPKI-1-CERT_EXPIRING_ALERT : Certificate expiring WITHIN A WEEK.
Trustpoint Name= check, Certificate Type= ID, Serial Number= 02:EC,
Issuer Name= CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN, Subject name= CN=cisco.com,
Time Left= 1 days, 23 hours, 59 minutes, 41 seconds
```

Restrictions for PKI Credentials Expiry Alerts

Alerts are not sent for the following certificates:

- Secure Unique Device Identifier (SUDI) certificates
- Certificates that belong to a trustpool. Trustpools have their own expiry alerts mechanism
- Trustpoint clones
- CA certificates that do not have a router certificate associated with it.
- Certificates with key usage keys

Enable PKI Traps

This feature cannot be disabled and requires no additional configuration tasks.

To enable PKI traps, use the **snmp-server traps pki** command. If SNMP is configured, the SNMP trap is configured in the same PKI expiry timer.

```
Router(config)# snmp-server traps pki
Router(config)# commit
```

Verification

This example shows sample output from the show running-config command.

```
Router# show runn snmp-server traps
snmp-server traps pki
```

What's Next: See Regenerate the Certificate.

Regenerate the Certificate

The certificate becomes invalid once expired. When you see the certificate expiry notification, we recommend you to regenerate the certificate, as soon as possible.

Perform the following steps, to regenerate the certificates:

1. Clear the existing certificate using the following command:

   ```
   Router# clear crypto ca certificates [trustpoint-name]
   ```

   For example,

   ```
   Router# clear crypto ca certificates myca
   ```

2. We recommend you to regenerate a new keypair for the label configured under the trustpoint-name. The new keypair overwrites the old key pair.

   ```
   Router# crypto key generate rsa [keypair-lablel]
   ```

   For example,

   ```
   Router# crypto key generate rsa mykey
   The name for the keys will be: mykey
   % You already have keys defined for mykey
   Do you really want to replace them? [yes/no]: yes
     Choose the size of the key modulus in the range of 512 to 4096 for your General Purpose
    Keypair. Choosing a key modulus greater than 512 may take a few minutes.

   How many bits in the modulus [2048]:
   ```

```
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]The name for the keys will be: mykey
% You already have keys defined for mykey
Do you really want to replace them? [yes/no]: yes
  Choose the size of the key modulus in the range of 512 to 4096 for your General Purpose
 Keypair. Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [2048]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

3. Reenroll the certificate using the following command. For more information, see Request Your Own Certificates, on page 111.

```
Router# crypto ca authenticate [trustpoint-name]
Router# crypto ca enroll [trustpoint-name]
```

For example,

```
Router# crypto ca authenticate myca
Router# crypto ca enroll myca
```

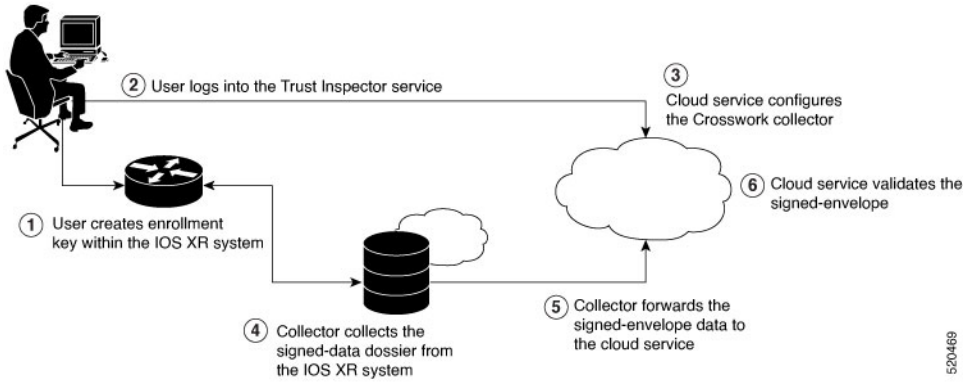
Integrating Cisco IOS XR and Crosswork Trust Insights

The Cisco IOS XR Software provides you the infrastructure to enroll and share the signed-data with Cisco Crosswork cloud infrastructure and applications. The Cisco Crosswork Trust Insights is a cloud-based Software as a service (SaaS) that provides signed and encrypted system integrity information to track the trust posture of network hardware and software components. For details, see Cisco Crosswork Trust Insights Data Sheet.

Integrating IOS XR and Crosswork Trust Insights include these main processes:

- System enrollment – Enrolling a Cisco IOS XR platform into Crosswork cloud infrastructure.
- Signed-data sharing – Sharing the data for infrastructure trust analysis between the systems that run IOS XR and Crosswork. This involves collecting the signed-data dossier, that is, signed-data that is needed for infrastructure trust inspection service.

Workflow



The following steps depict the workflow of Cisco IOS XR and Crosswork Trust Insights integration:

1. As part of the enrollment process, the user generates new key pair and trust root within the IOS XR system by using the IOS XR commands.
2. The user logs into the Trust Inspector service, and enters the enrollment workflow in the enrollment dialog to create a new device ID. The user must provide the management IP address, login credentials and certificate root to the Trust Inspector service.
3. The Trust Inspector service configures the Crosswork collector to log in to the router, and to pull the data that is pushed down from the cloud to the collector.
4. The Crosswork collector begins a periodic polling cycle and executes a command to generate a signed-information dossier from each IOS XR instance that is being polled.
5. The collector forwards the signed-envelope data to the cloud service for validation.
6. The cloud service validates signed-envelope against the enrolled certificate or trust chain.

How to Integrate Cisco IOS XR and Crosswork Trust Insights

Integrating Cisco IOS XR and Crosswork Trust Insights involve these main tasks for system enrollment and data-signing:

- Generate Key Pair, on page 124
- Generate System Trust Point for the Leaf and Root Certificate, on page 125
- Generate Root and Leaf Certificates, on page 126
- Collect Data Dossier, on page 129

Prerequisites

Before you begin, you must check here for any available IOS XR Software Maintenance Updates (SMUs) specific to Crosswork Trust Insights. For information related to SMUs, see Cisco IOS XR Release Notes.

You must ensure that the below configurations are present on the IOS XR device, before starting IOS XR and Crossworks Trust Insights integration.

- User authorization required to collect the signed-data dossier
- SSH server configuration
- Netconf server configuration
- Domain name configuration, which is required for certification enrollment

The sections given below lists the configuration example for the prerequisites.

Configuration Example for User Authorization

You must have the required user access privileges in order to collect the data dossier from the system. This is defined in terms of IOS XR Task IDs for each command.

For the respective Task ID applicable for each data dossier option and for the signed-envelope, see the Task ID section in the Command Reference page of **show platform security integrity dossier** command and **utility sign** command.

Note We recommend that you use the **task execute dossier** to configure a CTI (customer-define) user, who collects dossier from the system.

Listed below are the configurations to set up a user with sufficient authorization to collect all the signed-data dossier. You can configure customized task groups, then associate those task groups with user groups, and finally associate the user groups with the user.

```
Router#configure
Router(config)#taskgroup alltasks-dossier
Router(config-tg)#task read sysmgr
Router(config-tg)#task read system
Router(config-tg)#task read pkg-mgmt
Router(config-tg)#task read basic-services
Router(config-tg)#task read config-services
Router(config-tg)#task execute dossier
Router(config-tg)#task execute basic-services
Router(config-tg)#commit
```

```
Router#configure
Router(config)#usergroup dossier-group
Router(config-ug)#taskgroup alltasks-dossier
Router(config-ug)#commit
```

```
Router#configure
Router(config)#username dossier-user
Router(config-un)#group dossier-group
Router(config-un)#commit
```

Configuration Example for for SSH and Netconf

```
Router#configure
Router(config)#ssh server v2
Router(config)#ssh server vrf default
Router(config)#ssh server netconf vrf default
Router(config)#netconf-yang agent
Router(config-ncy-agent)#ssh
Router(config-ncy-agent)#exit
Router(config)#domain name example.com
Router(config)#commit
```

Running Configuration

```
ssh server v2
ssh server vrf default
ssh server netconf vrf default
!
netconf-yang agent
 ssh
!
domain name example.com
```

While the dossier is collected from a device through SSH, the SSH session might timeout. Also, multiple ssh sessions to a device can result in the denial of some SSH sessions. To avoid such occurrence, the following configuration is recommended on the device:

```
Router#configure
Router(config)#ssh server rate-limit 600
Router(config)#line default
Router(config-line)#exec-timeout 0 0
Router(config-line)#session-timeout 0
Router(config-line)#commit
```

Running Configuration

```
ssh server rate-limit 600
!
line default
 exec-timeout 0 0
 session-timeout 0
!
```

Generate Key Pair

To enroll a system running Cisco IOS XR Software, you must generate the key and the certificate for both the leaf and the root node. The system supports a two tier self-signed certificate chain for the enrollment key to support re-keying without re-enrollment of the certificate with the Crossworks service.

You can use the **system-root-key** and **system-enroll-key** options in the **crypto key generate** command to generate the root key and the enrollment key respectively, for all the hashing algorithms. You can do this for hashing algorithms such as RSA, DSA or ECDSA (including ECDSA nistp384 and ECDSA nitsp521).

Example of Generating Key Pair

Key pair generation for root:

```
Router#crypto key generate rsa system-root-key

Sun Oct 20 13:05:26.657 UTC
The name for the keys will be: system-root-key
Choose the size of the key modulus in the range of 512 to 4096 for your General Purpose
Keypair. Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus [2048]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

Key pair generation for leaf:

```
Router#crypto key generate rsa system-enroll-key

Sun Oct 20 13:05:40.370 UTC
The name for the keys will be: system-enroll-key
Choose the size of the key modulus in the range of 512 to 4096 for your General Purpose
Keypair. Choosing a key modulus greater than 512 may take a few minutes.
 How many bits in the modulus [2048]:
Generating RSA keys ...
```

```
Done w/ crypto generate keypair
[OK]
```

Verification

You can use the **show crypto key mypubkey rsa** command to verify the above key pair generation.

```
Router#show  crypto  key mypubkey rsa | begin system-
Fri Mar 27 14:00:20.954 IST
Key label: system-root-key
Type     : RSA General purpose
Size     : 2048
Created  : 01:13:10 IST Thu Feb 06 2020
Data     :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
 B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
 F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
 39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
 AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
 9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
 EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
 22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
 11020301 0001

Key label: system-enroll-key
Type     : RSA General purpose
Size     : 2048
Created  : 01:13:16 IST Thu Feb 06 2020
Data     :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 009DBC14 C83604E4 EB3D3CF8 5BA7FDDB 80F7E85B 427332D8 BBF80148 F0A9C281
 49F87D5C 0CEBA532 EBE797C5 7F174C69 0735D13A 493670CB 63B04A12 4BCA7134
 EE0031E9 047CAA1E 802030C5 6071E8C2 F8ECE002 CC3B54E7 5FD24E5C 61B7B7B0
 68FA2EFA 0B83799F 77AE4621 435D9DFF 1D713108 37B614D3 255020F9 09CD32E8
 82B07CD7 01A53896 6DD92B5D 5119597C 98D394E9 DBD1ABAF 6DE949FE 4A8BF1E7
 851EB3F4 60B1114A 1456723E 063E50C4 2D410906 BDB7590B F1D58480 F3FA911A
 6C9CD02A 58E68D04 E94C098F 0F0E81DB 76B40C55 64603499 2AC0547A D652412A
 BCBBF69F 76B351EE 9B2DF79D E490C0F6 92D1BB97 B905F33B FAB53C20 DDE2BB22
 C7020301 0001
```

Associated Commands

- crypto key generate dsa
- crypto key generate ecdsa
- crypto key generate rsa
- show crypto key mypubkey dsa
- show crypto key mypubkey ecdsa
- show crypto key mypubkey rsa

Generate System Trust Point for the Leaf and Root Certificate

You must configure these steps to generate the system trust point for the root and the leaf certificate:

Configuration Example

```
Router#config
Router(config)#domain name domain1
Router(config)#crypto ca trustpoint system-trustpoint
Router(config)#keypair rsa system-enroll-key
Router(config)#ca-keypair rsa system-root-key
Router(config)#subject-name CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
Router(config)#subject-name ca-certificate CN=lab1-ca,C=US,ST=CA,L=San Jose,O=cisco
systems,OU=ASR
Router(config)#enrollment url self
Router(config)#key-usage certificate digitalsignature keyagreement dataencipherment
Router(config)#lifetime certificate 300
Router(config)#message-digest sha256
Router(config)#key-usage ca-certificate digitalsignature keycertsign crlsign
Router(config)#lifetime ca-certificate 367
Router(config)#commit
```

Running Configuration

```
config
domain name domain1
crypto ca trustpoint system-trustpoint
keypair rsa system-enroll-key
ca-keypair rsa system-root-key
subject-name CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
subject-name ca-certificate CN=lab1-ca,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
enrollment url self
key-usage certificate digitalsignature keyagreement dataencipherment
lifetime certificate 300
message-digest sha256
key-usage ca-certificate digitalsignature keycertsign crlsign
lifetime ca-certificate 367
!
```

Associated Commands

- ca-keypair
- crypto ca trustpoint
- domain
- enrollment
- key-usage
- key-pair
- lifetime
- message-digest
- subject-name

Generate Root and Leaf Certificates

You must perform these steps to generate the root and the leaf certificates.

The root certificate is self-signed. The root certificate signs the leaf certificate.

Example of Generating Root Certificate

```
Router#crypto ca authenticate system-trustpoint

Sun Oct 20 13:07:24.136 UTC
% The subject name in the certificate will include: CN=lab1
ca,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
% The subject name in the certificate will include: ios.cisco.com
Serial Number  : 0B:62
Subject:
serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco systems,L=San
Jose,ST=CA,C=US,CN=lab1-ca
Issued By      :
        serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco systems,L=San
Jose,ST=CA,C=US,CN=lab1-ca
  Validity Start : 13:07:26 UTC Sun Oct 20 2019
  Validity End   : 13:07:26 UTC Wed Oct 21 2020
  SHA1 Fingerprint:
         9DD50A6B24FEBC1DDEE40CD2B4D99A829F260967
```

Example of Generating Leaf Certificate

```
Router#crypto ca enroll system-trustpoint

Sun Oct 20 13:07:45.593 UTC
% The subject name in the certificate will include: CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco
 systems,OU=ASR
% The subject name in the certificate will include: ios.cisco.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: c44a11fc
% Include an IP address in the subject name? [yes/no]: no
Certificate keypair configured Type: 1, Label: system-enroll-key.Leaf cert key usage string:
 critical,digitalSignature,keyEncipherment,keyAgreement.  Serial Number  : 0B:63
  Subject:
        serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco systems,L=San
Jose,ST=CA,C=US,CN=lab1-ads
  Issued By      :
        serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco systems,L=San
Jose,ST=CA,C=US,CN=lab1-ca
  Validity Start : 13:07:47 UTC Sun Oct 20 2019
  Validity End   : 13:07:47 UTC Sat Aug 15 2020
  SHA1 Fingerprint:
         19D4C40F9EFF8FF25B59DE0161BA6C0706DC9E3A
```

Verification

You can use the **show crypto ca certificates system-trustpoint [detail]** command to see the details of generated root and leaf certificates:

```
Router#show crypto ca certificates system-trustpoint
Fri Mar 27 14:00:51.037 IST

Trustpoint       : system-trustpoint
================================================
CA certificate
  Serial Number  : 10:B5
  Subject:
        serialNumber=7b20faa4,unstructuredName=test-sec1.cisco.com
```

```
  Issued By      :
        serialNumber=7b20faa4,unstructuredName=test-sec1.cisco.com
  Validity Start : 12:30:17 UTC Fri Feb 21 2020
  Validity End   : 12:30:17 UTC Sat Feb 20 2021
  SHA1 Fingerprint:
         9400A30816805219FAAA5B9C86C214E6F34CEF7B
Router certificate
  Key usage      : General Purpose
  Status         : Available
  Serial Number  : 10:B6
  Subject:

serialNumber=7b20faa4,unstructuredAddress=10.1.1.1,unstructuredName=test-sec1.cisco.com,CN=Anetwork,OU=IT,O=Spark
 Network,L=Rotterdam,ST=Zuid Holland,C=NL
  Issued By      :
        serialNumber=7b20faa4,unstructuredName=test-sec1.cisco.com
  Validity Start : 12:30:31 UTC Fri Feb 21 2020
  Validity End   : 12:30:31 UTC Sat Feb 20 2021
  SHA1 Fingerprint:
         21ACDD5EB6E6F4103E02C1BAB107AD86DDCDD1F3
Associated Trustpoint: system-trustpoint
```

Associated Commands

- **crypto ca authenticate**
- **crypto ca enroll**
- **show crypto ca certificates system-trustpoint**

System Certificates Expiry

You need to regenerate the certificate, before it expires. From Release 7.1.1, IOS -XR provides a mechanism by which a CA client sends a notification to a syslog server when certificates are on the verge of expiry. For more information see Learn About the PKI Alert Notification, on page 119.

When you see the certificate expiry notification, we recommend you to regenerate the certificate, see Regenerate the Certificate, on page 120.

The following example shows how to regenerate the certificate.

```
Router# clear crypto ca certificates system-trustpoint
Router# crypto ca authenticate system-trustpoint
Router# crypto ca enroll system-trustpoint
```

Collect Data Dossier

Table 13: Feature History Table

Feature Name	Release Information	Description
Collect Filesystem Inventory	Release 7.3.1	With this feature, a snapshot of the filesystem metadata such as when the file was created, modified, or accessed is collected at each configured interval. In addition to displaying the changes that the file underwent as compared to the previous snapshot, the inventory helps in maintaining data integrity of all the files in the system.
IMA Optimization	Release 7.3.1	Integrity Measurement Architecture (IMA) is a Linux-based utility that attests and appraises the integrity of a system security, at runtime. In this release, IMA introduces the following IMA optimization aspects: • Incremental IMA that collects IMA events selectively and progressively instead of collecting all the IMA events at the same time. You can define the start of an IMA sequence, which consists of start event, start sequence number, and start time. • SUDI Signature - provides the hardware root of trust to the dossier that is collected by the system.

Feature Name	Release Information	Description
Support for Display Compact Option	Release 7.4.1	This release introduces: • Display compact option in the dossier CLI, thereby allowing you to obtain IMA event logs in the **protobuf** format, which can be decoded at a client site. This provides flexibility to use any decoding mechanism Use the `display compact` keyword with the existing `show platform security integrity dossier include system-integrity-snapshot` command.

The Cisco IOS XR Software provides a data dossier command, **show platform security integrity dossier**, that helps in collecting the data from various IOS XR components. The output is presented in JSON format.

You can choose various selectors for this command as given below :

```
Router#show platform security integrity dossier include packages reboot-history
rollback-history system-integrity-snapshot system-inventory nonce 1580 | utility sign nonce
 1580 include-certificate
```

Create Signed-Envelope

To verify the data integrity and authenticity of the data dossier output, a signature is added to the output data. To enable this feature, you can use the **utility sign** command along with the **show platform security integrity dossier** command. The output is presented in JSON format.

This **utility sign** can also be used with any of the IOS XR commands.

Note The Secure Unique Device Identifier or SUDI signature provides the hardware root of trust to the dossier that is collected by the system.

Verification Example

```
Router#show  platform security integrity  dossier include  reboot-history nonce 1580 |
utility sign nonce 1580 include-certificate
NCS560
```

Collect Filesystem Inventory

The metadata of the filesystem can be collected using data dossier. The metadata of the file includes information about time the file was created, last accessed, last modified and so on. A snapshot is captured at each configured

interval. The initial snapshot shows a complete snapshot of all files in the filesystem. The files are scanned periodically and new inventory data is collected and stored as incremental snapshots.

To enable this feature, use the **filesystem-inventory** command.

```
Router(config)#filesystem-inventory
Router(config-filesystem-inventory)#snapshot-interval 2
Router(config-filesystem-inventory)#commit
```

The `snapshot-interval` is the time interval in 15-minute blocks. The interval ranges 1–96. For example, value of `2` indicates that a snapshot interval is collected every 30 minutes. The snapshots are stored in`./misc/scratch/filesysinv` The logs are stored in `/var/log/iosxr/filesysinv/*`.

To retrieve the filesystem inventory, use the following dossier command. Output is presented in JSON format.

```
show platform security integrity dossier include filesystem-inventory | file
<platform>-parent.json
```

```
{"collection-start-time":1610168028.380901,
"model-name":"http://cisco.com/ns/yang/Cisco-IOS-XR-ama",
"model-revision":"2019-08-05","license-udi":{"result-code": "Success", "license-udi":
"UDI: PID:NCS-55A1-24H,SN:FOC2104R15R\n"},"version":{"result-code": "Success",
"version": "Cisco IOS XR Software, Version 7.3.1
\nCopyright (c) 2013-2020 by Cisco Systems, Inc.\n\nBuild Information:\n
Built By     : <user>\n Built On     : Thu Jan  7 17:16:02 PST 2021\n
Built Host   : <host>\n Workspace    : <ws>
Version     : 7.3.1\n  Location    : /opt/cisco/XR/packages/\n Label       : 7.3.1\n\ncisco

() processor\nSystem uptime is 8 hours 7 minutes\n\n"},"platform":{"result-code":
"Success", "platform":
"Node              Type                         State             Config state
-------------------------------------------------------------------------------
0/RP0/CPU0        <node-type>(Active)       IOS XR RUN        NSHUT\n
0/RP0/NPU0        Slice                      UP
0/RP0/NPU1        Slice                      UP
0/FT0             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
0/FT1             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
0/FT2             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
PM1               <platform>-1100W-ACRV      OPERATIONAL       NSHUT
"},
----------------------------Output is snipped for brevity
-------------------------------------
```

To limit the number of snapshots, use the following command:

```
show platform security integrity dossier include filesystem-inventory
filesystem-inventory-options '{\"0/RP0/CPU0\": {\"block_start\": 0, \"count\": 1}}'
```

To start from a new block, use the following command:

```
show platform security integrity dossier include filesystem-inventory
filesystem-inventory-options '{\"0/RP0/CPU0\": {\"block_start\": 5}}'
```

To collect data from a remote node, use the following command:

```
show platform security integrity dossier include filesystem-inventory
filesystem-inventory-options '{\"0/RP1/CPU0\": {\"block_start\": 0}}' | file
harddisk:PE1_remote.json
```

Following is the sample of the display compact container:

```
{"node-data":[{"node-location":"node0_RP0_CPU0","up-time":150311,"start-time":"Tue Jul 27
13:55:12 2021","ima-event-log-compact":
["IlYIABoMCO+ggIgGEKmxwZYBIkQIABAKGhTU2yPVDA5Rx+64ecp41qZQrLEWSCACKhSXl+340O7Ta
xz5JUeBYFHIr05F7jIOYm9vdF9hZ2dyZWdhdGVAAQ=="]}]}
```

Incremental Integrity Measurement Architecture

With incremental Integrity Meansurement Architecture (IMA), you can define the starting IMA sequence that you want to include in a response. The system then starts to report the subsequent events.

```
show platform security integrity dossier incremental-ima
"{\"ima_start\":[{\"0/RP0/CPU0\":{\"start_event\":1000,\"start_time\":\"Tue Feb 16 09:15:17
 2021\"}}]}"
```

Associated Command

- **show platform security integrity dossier**
- **utility sign**

Procedure to Test Key Generation and Data-signing with Different Key Algorithm

You can follow these steps to test key generation and data-signing with a different key algorithm:

- Unconfigure the trustpoint (using the **no crypto ca trustpoint system-trustpoint** command)
- Clear the certificates that were generated earlier (using the **clear crypto ca certificates system-trustpoint** command)
- Generate new keys.
- Configure the system trustpoint again.
- Authenticate and enroll the system trustpoint to generate the certificates.

See How to Integrate Cisco IOS XR and Crosswork Trust Insights, on page 122 section for configuration steps of each task.

Verify Authenticity of RPM Packages Using Fingerprint

Table 14: Feature History Table

Feature Name	Release Information	Description
Verify Authenticity of RPM Packages Using Fingerprint	Release 7.3.1	This feature helps in verifying the authenticity of an installable package using fingerprint values. The fingerprint value of the package is compared with a point of reference called Known Good Value (KGV). The KGV for an image or package is generated after it is built by Cisco. After installing the package, the associated install time and build time fingerprint values are compared using Yang RPC to determine whether the package is genuine. A match in the fingerprints indicates that the package published on CCO and that installed on router are the same.

Is there a simple way to determine the authenticity of a package that is installed on a router? Is there a mechanism to identify whether a package signature is checked at install time, or detect changes to the files after the package is installed at run time?

Cisco IOS XR, Release 7.3.1 introduces a fingerprint mechanism to verify the authenticity of a package that Cisco releases. This mechanism helps determine whether the installed package is genuine, where the installed and running software matches the software that is published by Cisco.

There are significant security measures for installing software using GPG and IMA signing. However, there is need to report more data for Cisco Crosswork application to monitor and flag potential issues for further investigation. Cisco Crosswork monitors the installed software over a period to help accomplish the following tasks:

- To determine whether there are any differences between the software that is published on Cisco.com and that downloaded to the router.
- To determine whether any files in a package have been altered, either accidentally or maliciously, from the time the package was installed.

A Known Good Value (KGV) is calculated and published for each package. This value is considered the right value for the package.

Two fingerprint (hex) values for each active or committed packages are monitored to ensure authenticity of the package:

- **Install time fingerprint:** Hex value that represents the software in the package at install time. An RPM is genuine if it is not modified before install, and it matches the KGV. Whereas a manipulated RPM shows a mismatch in the fingerprint that is published in the KGV.
- **Run time fingerprint:** Hex value that represents the running software of an installed package. The value matches the corresponding install time fingerprint if the RPM has not been modified since the install time. If there are changes to the files, the run time and install time fingerprints show a mismatch. Every time the files that are installed by an RPM are changed, the run time fingerprint also changes. A value of `0` (zero) is displayed if no run time fingerprint is available for a package. This is used to monitor changes to the running software over time.



Note These two values are displayed only in the Yang model output. No CLI commands are provided to view these values.

```
Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:97f5bc36-0eb0-4d2f-9c6f-3d34fea14be0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
    <install xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spirit-install-instmgr-oper">
      <packages>
        <active>
         <summary>
          <rpm-fingerprint-status>generation-up-to-date</rpm-fingerprint-status>
           <rpm-fingerprint-timestamp>Mon Jun 15 15:58:22 2020</rpm-fingerprint-timestamp>

            <package>
             <name>asr9k-xr</name>
              <version>7.3.1</version>
```

```
              <release>r731</release>
             <gpg-key-id>ddcead3dcb38048d</gpg-key-id>
              <rpm-fingerprint>

<rpm-fingerprint-install-time>2871bf68d3cd764938775afc9e5a69c130f9fbde</rpm-fingerprint-install-time>


<rpm-fingerprint-run-time>2871bf68d3cd764938775afc9e5a69c130f9fbde</rpm-fingerprint-run-time>

              </rpm-fingerprint>
             </package>

            <package>
             <name>asr9k-mcast-x64</name>
             <version>2.0.0.0</version>
             <release>r731</release>
             <gpg-key-id>ddcead3dcb38048d</gpg-key-id>
              <rpm-fingerprint>

<rpm-fingerprint-install-time>3ddca55bc00a0ce2c2e52277919d398621616b28</rpm-fingerprint-install-time>


<rpm-fingerprint-run-time>3ddca55bc00a0ce2c2e52277919d398621616b28</rpm-fingerprint-run-time>

              </rpm-fingerprint>
             </package>
------------------------------ Truncated for brevity ---------------------------------
```

In the example, both the install time and run time fingerprints are the same.

The fingerprint generation status is used to indicate how up-to-date the run time fingerprints are. This may indicate that generation is currently in progress and will complete shortly, or generation is awaiting the end of an atomic change.

Support for Ed25519 Public-Key Signature System

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Support for Ed25519 Public-Key Signature System	Release 7.3.1	This feature allows you to generate and securely store crypto key pair for the Ed25519 public-key signature algorithm on Cisco IOS XR 64-bit platforms. This signature system provides fast signing, fast key generation, fool proof session keys, collision resilience, and small signatures. The feature also facilitates integration of Cisco IOS XR with Cisco Crosswork Trust Insights. Commands introduced for this feature are: crypto key generate ed25519 crypto key zeroize ed25519 show crypto key mypubkey ed25519 Commands modified for this feature are: ca-keypair keypair

The Cisco IOS XR Software Release 7.3.1 introduces the support for Ed25519 public-key signature algorithm on 64-bit platforms. Prior to this release, only DSA, ECDSA, and RSA signature algorithms were supported. The Ed25519 signature algorithm uses the elliptic curve cryptography that offers a better security with faster performance when compared to other signature algorithms.

You can generate the Ed25519 crypto keys either with an empty label or with two predefined labels: **system-root-key** and **system-enroll-key**. In the case of an empty label, the system generates the key pair against the default label. You can use the key pairs with the predefined labels to integrate Cisco IOS XR with Cisco Crosswork Trust Insights.

Generate Crypto Key for Ed25519 Signature Algorithm

Configuration Example

To generate the Ed25519 crypto key, use the **crypto key generate ed25519** command in XR EXEC mode.

```
Router#crypto key generate ed25519
```

To delete the Ed25519 crypto key with default label or any predefined label, use the **crypto key zeroize ed25519** command in XR EXEC mode.

Verification

Use the **show crypto key mypubkey ed25519** command to view all Ed25519 crypto keys generated on the system.

```
Router# show crypto key mypubkey ed25519

Mon Nov 30 07:05:06.532 UTC
Key label: the_default
Type : ED25519
Size : 256
Created : 07:03:17 UTC Mon Nov 30 2020
Data :
FF0ED4E7 71531B3D 9ED72C48 3F79EC59 9EFECCC3 46A129B2 FAAA12DD EE9D0351
```

Related Topics

- Support for Ed25519 Public-Key Signature System, on page 135

Associated Commands

- **crypto key generate ed25519**
- **crypto key zeroize ed25519**
- **show crypto key mypubkey ed25519**

Information About Implementing Certification Authority

Supported Standards for Certification Authority Interoperability

Cisco supports the following standards:

- IKE—A hybrid protocol that implements Oakley and Skeme key exchanges inside the Internet Security Association Key Management Protocol (ISAKMP) framework. Although IKE can be used with other protocols, its initial implementation is with the IPSec protocol. IKE provides authentication of the IPSec peers, negotiates IPSec keys, and negotiates IPSec security associations (SAs).
- Public-Key Cryptography Standard #7 (PKCS #7)—A standard from RSA Data Security Inc. used to encrypt and sign certificate enrollment messages.
- Public-Key Cryptography Standard #10 (PKCS #10)—A standard syntax from RSA Data Security Inc. for certificate requests.
- RSA keys—RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Adelman. RSA keys come in pairs: one public key and one private key.
- SSL—Secure Socket Layer protocol.
- X.509v3 certificates—Certificate support that allows the IPSec-protected network to scale by providing the equivalent of a digital ID card to each device. When two devices want to communicate, they exchange

digital certificates to prove their identity (thus removing the need to manually exchange public keys with each peer or specify a shared key at each peer). These certificates are obtained from a CA. X.509 as part of the X.500 standard of the ITU.

Certification Authorities

Purpose of CAs

CAs are responsible for managing certificate requests and issuing certificates to participating IPSec network devices. These services provide centralized key management for the participating devices.

CAs simplify the administration of IPSec network devices. You can use a CA with a network containing multiple IPSec-compliant devices, such as routers.

Digital signatures, enabled by public key cryptography, provide a means of digitally authenticating devices and individual users. In public key cryptography, such as the RSA encryption system, each user has a key pair containing both a public and a private key. The keys act as complements, and anything encrypted with one of the keys can be decrypted with the other. In simple terms, a signature is formed when data is encrypted with a user's private key. The receiver verifies the signature by decrypting the message with the sender's public key. The fact that the message could be decrypted using the sender's public key indicates that the holder of the private key, the sender, must have created the message. This process relies on the receiver's having a copy of the sender's public key and knowing with a high degree of certainty that it does belong to the sender and not to someone pretending to be the sender.

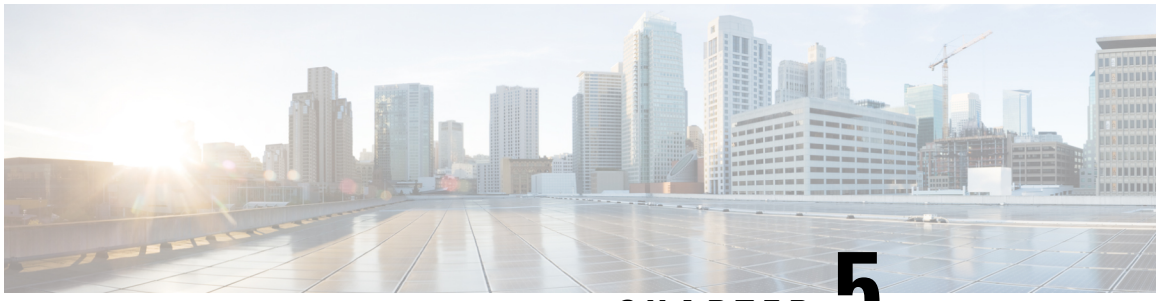
Digital certificates provide the link. A digital certificate contains information to identify a user or device, such as the name, serial number, company, department, or IP address. It also contains a copy of the entity's public key. The certificate is itself signed by a CA, a third party that is explicitly trusted by the receiver to validate identities and to create digital certificates.

To validate the signature of the CA, the receiver must first know the CA's public key. Normally, this process is handled out-of-band or through an operation done at installation. For instance, most web browsers are configured with the public keys of several CAs by default. IKE, an essential component of IPSec, can use digital signatures to authenticate peer devices for scalability before setting up SAs.

Without digital signatures, a user must manually exchange either public keys or secrets between each pair of devices that use IPSec to protect communication between them. Without certificates, every new device added to the network requires a configuration change on every other device with which it communicates securely. With digital certificates, each device is enrolled with a CA. When two devices want to communicate, they exchange certificates and digitally sign data to authenticate each other. When a new device is added to the network, a user simply enrolls that device with a CA, and none of the other devices needs modification. When the new device attempts an IPSec connection, certificates are automatically exchanged and the device can be authenticated.

CA Registration Authorities

Some CAs have a registration authority (RA) as part of their implementation. An RA is essentially a server that acts as a proxy for the CA so that CA functions can continue when the CA is offline.



CHAPTER 5

Implementing Keychain Management

This module describes how to implement keychain management on. Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.

- Implementing Keychain Management, on page 139

Implementing Keychain Management

This module describes how to implement keychain management on. Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.

Restrictions for Implementing Keychain Management

You must be aware that changing the system clock impacts the validity of the keys in the existing configuration.

Configure Keychain

This task configures a name for the keychain.

You can create or modify the name of the keychain.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
RP/0/RP0/CPU0:router(config-isis-keys)#
```

Creates a name for the keychain.

Note
Configuring only the keychain name without any key identifiers is considered a nonoperation. When you exit the configuration, the router does not prompt you to commit changes until you have configured the key identifier and at least one of the mode attributes or keychain-key configuration mode attributes (for example, lifetime or key string).

Step 3 **commit**

Commits the configuration changes and remains within the configuration session.

Step 4 **show key chain** *key-chain-name*

Example:

```
RP/0/RP0/CPU0:router# show key chain isis-keys

Key-chain: isis-keys/ -

accept-tolerance -- infinite
Key 8 -- text "1104000E120B520005282820"
  cryptographic-algorithm -- MD5
  Send lifetime:   01:00:00, 29 Jun 2006 - Always valid  [Valid now]
  Accept lifetime: 01:00:00, 29 Jun 2006 - Always valid [Valid now]
```

(Optional) Displays the name of the keychain.

Note
The *key-chain-name* argument is optional. If you do not specify a name for the *key-chain-name* argument, all the keychains are displayed.

Step 5 **show run**

Example:

```
key chain isis-keys
 accept-tolerance infinite
 key 8
  key-string mykey91abcd
  cryptographic-algorithm MD5
  send-lifetime 1:00:00 june 29 2006 infinite
  accept-lifetime 1:00:00 june 29 2006 infinite
  !
 !
!
```

Configure Tolerance Specification to Accept Keys

This task configures the tolerance specification to accept keys for a keychain to facilitate a hitless key rollover for applications, such as routing and management protocols.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0//CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **accept-tolerance** *value* [**infinite]**

Example:

```
RP/0//CPU0:router(config-isis-keys)# accept-tolerance infinite
```

Configures an accept tolerance limit—duration for which an expired or soon-to-be activated keys can be used for validating received packets—for a key that is used by a peer.

- Use the *value* argument to set the tolerance range in seconds. The range is from 1 to 8640000.
- Use the **infinite** keyword to specify that an accept key is always acceptable and validated when used by a peer.

Step 4 **commit**

Commits the configuration changes and remains within the configuration session.

Configure Key Identifier for Keychain

This task configures a key identifier for the keychain.

You can create or modify the key for the keychain.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0//CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
RP/0//CPU0:router(config-isis-keys)# key 8
```

Creates a key for the keychain. The key ID has to be unique within the specific keychain.

- Use the *key-id* argument as a 48-bit integer.

Step 4 **commit**

Commits the configuration changes and remains within the configuration session.

Configure Text for Key String

This task configures the text for the key string.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0//CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
RP/0//CPU0:router(config-isis-keys)# key 8
RP/0//CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **key-string** [**clear** | **password**] *key-string-text*

Example:

```
RP/0//CPU0:router(config-isis-keys-0x8)# key-string password 8
```

Specifies the text string for the key.

- Use the **clear** keyword to specify the key string in clear text form; use the **password** keyword to specify the key in encrypted form.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Determine Valid Keys

This task determines the valid keys for local applications to authenticate the remote peers.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
```

Creates a a name for the keychain.

Step 3 **key** *key-id*

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/RP0/CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **accept-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/(config-isis-keys-0x8)# accept-lifetime 1:00:00 october 24 2005 infinite
```

(Optional) Specifies the validity of the key lifetime in terms of clock time. You can specify the *start-time* and *end-time* in *hh:mm:ss month DD YYYY* format or *hh:mm:ss DD month YYYY* format.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure Keys to Generate Authentication Digest for Outbound Application Traffic

This task configures the keys to generate authentication digest for the outbound application traffic.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/RP0/CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **send-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)#key 8
RP/0/(config-isis-keys-0x8)# send-lifetime 1:00:00 october 24 2005 infinite
```

(Optional) Specifies the set time period during which an authentication key on a keychain is valid to be sent. You can specify the validity of the key lifetime in terms of clock time.

In addition, you can specify a start-time value and one of the following values:

- **duration** keyword (seconds)
- **infinite** keyword
- *end-time* argument

If you intend to set lifetimes on keys, Network Time Protocol (NTP) or some other time synchronization method is recommended.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure Cryptographic Algorithm

This task allows the keychain configuration to accept the choice of the cryptographic algorithm.

From Cisco IOS XR Software Release 7.1.2 Release 7.2.1 and later, you must follow the below guidelines while configuring the key chain. These are applicable only for FIPS mode (that is, when **crypto fips-mode** is configured).

- You must configure the session with a FIPS-approved cryptographic algorithm. A session configured with non-approved cryptographic algorithm for FIPS (such as, **MD5** and **HMAC-MD5**) does not work. This is applicable for OSPF, BGP, RSVP, ISIS, or any application using key chain with non-approved cryptographic algorithm.
- If you are using any **HMAC-SHA** algorithm for a session, then you must ensure that the configured *key-string* has a minimum length of 14 characters. Otherwise, the session goes down.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
RP/0/RP0/CPU0:router(config-isis-keys)#
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/RP0/CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **cryptographic-algorithm** [**HMAC-MD5** | **HMAC-SHA1-12** | **HMAC-SHA1-20** | **MD5** | **SHA-1** | **AES-128-CMAC-96** | **HMAC-SHA-256** | **HMAC-SHA1-96**]

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys-0x8)# cryptographic-algorithm MD5
```

Specifies the choice of the cryptographic algorithm. You can choose from the following list of algorithms:

- HMAC-MD5
- HMAC-SHA1-12
- HMAC-SHA1-20
- MD5
- SHA-1
- HMAC-SHA-256
- HMAC-SHA1-96
- AES-128-CMAC-96

The routing protocols each support a different set of cryptographic algorithms:

- Border Gateway Protocol (BGP) supports HMAC-MD5, HMAC-SHA1-12, HMAC-SHA1-96 and AES-128-CMAC-96.

- Intermediate System-to-Intermediate System (IS-IS) supports HMAC-MD5, SHA-1, MD5, AES-128-CMAC-96, HMAC-SHA-256, HMAC-SHA1-12, HMAC-SHA1-20, and HMAC-SHA1-96.
- Open Shortest Path First (OSPF) supports MD5, HMAC-MD5, HMAC-SHA-256, HMAC-SHA1-12, HMAC-SHA1-20, and HMAC-SHA1-96.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Lifetime of Key

If you are using keys as the security method, you must specify the lifetime for the keys and change the keys on a regular basis when they expire. To maintain stability, each party must be able to store and use more than one key for an application at the same time. A keychain is a sequence of keys that are collectively managed for authenticating the same peer, peer group, or both.

Keychain management groups a sequence of keys together under a keychain and associates each key in the keychain with a lifetime.



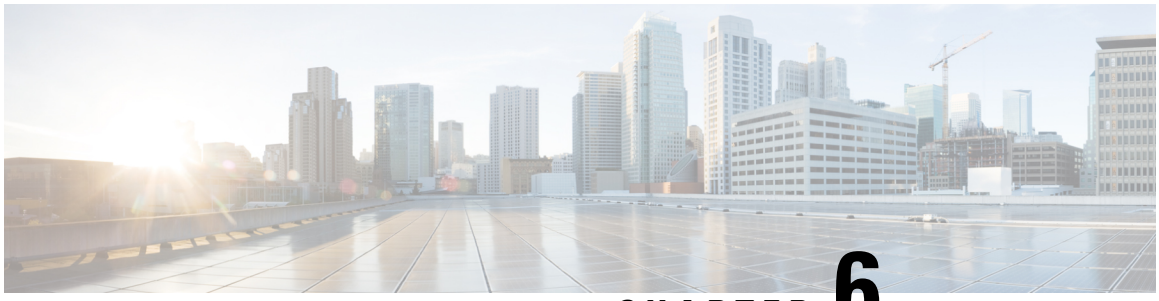
Note Any key that is configured without a lifetime is considered invalid; therefore, the key is rejected during configuration.

The lifetime of a key is defined by the following options:

- Start-time—Specifies the absolute time.
- End-time—Specifies the absolute time that is relative to the start-time or infinite time.

Each key definition within the keychain must specify a time interval for which that key is activated; for example, lifetime. Then, during a given key's lifetime, routing update packets are sent with this activated key. Keys cannot be used during time periods for which they are not activated. Therefore, we recommend that for a given keychain, key activation times overlap to avoid any period of time for which no key is activated. If a time period occurs during which no key is activated, neighbor authentication cannot occur; therefore, routing updates can fail.

Multiple keychains can be specified.



CHAPTER 6

Implementing Type 6 Password Encryption

Type 6 password encryption uses a reversible 128-bit AES encryption algorithm for storing passwords. Type 6 password encryption allows secure, and encrypted storage of plain-text passwords on the device. The device can decrypt the encrypted passwords into their original plain-text format.

You can use Type 6 password encryption to securely store plain text key strings for authenticating BGP, IP SLA, IS-IS, MACsec, OSPF, and RSVP sessions.

Feature History for Implementing Type 6 Password Encryption

Release	Modification
Release 7.0.1	This feature was introduced.

• How to Implement Type 6 Password Encryption , on page 149

How to Implement Type 6 Password Encryption

Scenario - The following 3-step process explains the Type 6 password encryption process for authenticating BGP sessions between two routers, R1 and R2.

Follow the first two steps for all Type 6 password encryption scenarios. The third step, *Creating BGP Sessions*, is specific to BGP. Similarly, you can enable Type 6 password encryption for OSPF, IS-IS, or other protocol sessions. For details on creating these protocol sessions, see the content in *Configure>Routing* listed here. For MACsec authentication, refer the **Configure MACsec** chapter.

Enabling Type6 Feature and Creating a Primary Key (Type 6 Server)

The Type6 encryption key, hereafter referred to as primary key in this chapter, is the password or key that encrypts all plain text key strings in the router configuration. An Advance Encryption Standard (AES) symmetric cipher does the encryption. The router configuration does not store the primary key. You cannot see or access the primary key when you connect to the router.

Creating the Primary Key

Use the **key config-key password-encryption** command to create the primary key.

Configuration Example

```
R1 & R2 # key config-key password-encryption

Fri Jul 19 12:22:45.519 UTC
New password Requirements: Min-length 6, Max-length 64
Characters restricted to [A-Z][a-z][0-9]
Enter new key :
Enter confirm key :
Master key operation is started in background
```

Once the command is executed, the **Master key operation**—creating, updating, or deleting the primary key—happens in the background. You can use the **show type6 server** command to view the status of the primary key operation.

When the key is created, it is stored internally; not as part of the router configuration. The router does not display the primary key as part of the running configuration. So, you cannot see or access the primary key when you connect to the router.

Enabling Type 6 Password Encryption

```
/* Enable Type 6 password encryption */
R1 & R2 (config)# password6 encryption aes
R1 & R2 (config)# commit
Fri Jul 19 12:22:45.519 UTC
```

Modifying the Primary Key

Note The Type 6 primary key update results in configuration change of the key chain and the other clients using Type 6. As the failure of router being configured can disrupt the product network, it is recommended to perform the primary key update operation during a maintenance window. Else, routing protocol sessions might fail.

The primary key is not saved to the running configuration, but the changes are persistent across reloads. The primary key update cannot be rolled back. That is, once the primary key is modified, you cannot revert to the older key using the **rollback configuration** command.

Enter the **key config-key password-encryption** command, and the old key and new key information.

```
R1 & R2# key config-key password-encryption

New password Requirements: Min-length 6, Max-length 64
Characters restricted to [A-Z][a-z][0-9]
Enter old key :
Enter new key :
Enter confirm key :
Master key operation is started in background
```

Deleting the Primary Key

```
R1 & R2# configure
R1 & R2 (config)# no password6 encryption aes
R1 & R2 (config)# commit
R1 & R2 (config)# exit
R1 & R2# key config-key password-encryption delete

WARNING: All type 6 encrypted keys will become unusable
Continue with master key deletion ? [yes/no]:yes
Master key operation is started in background
```

Verification

Verify that the primary key configuration and Type 6 feature configuration state are in the *Enabled* state. The **Master key Inprogress** field displays **No** to indicate that the primary key activity is complete (created, modified, or deleted). When you disable a primary key, **Disabled** is displayed for all the three states.

```
R1 & R2#show type6 server

Fri Jul 19 12:23:49.154 UTC
Server detail information:
==========================================
AES config State       :       Enabled
Masterkey config State :       Enabled
Type6 feature State    :       Enabled
Master key Inprogress  :       No
```

Verify Type 6 trace server details.

```
R1 & R2#show type6 trace server all

Fri Jul 19 12:26:05.111 UTC
Client file lib/type6/type6_server_wr
25 wrapping entries (18496 possible, 64 allocated, 0 filtered, 25 total)
Jul 19 09:59:27.168 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 ***** Type6 server process
started Respawn count (1) ****
…
…
Jul 19 12:22:59.908 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 User has started Master key
 operation (CREATE)
Jul 19 12:22:59.908 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Created Master key in TAM
successfully
Jul 19 12:23:00.265 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Master key Available set to
 (AVAILABLE)
Jul 19 12:23:00.272 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Master key inprogress set
to (NOT INPROGRESS)
```

From Cisco IOS XR Software Release 7.0.2 and later, you can use the **show type6 masterkey update status** command to display the update status of the primary key. Prior to this release, you could use the **show type6 clients** command for the same purpose.

```
Router#show type6 masterkey update status
Thu Sep 17 06:48:56.595 UTC
Type6 masterkey operation is NOT inprogress
```

```
Router#show type6 masterkey update status
Thu Sep 17 06:50:07.980 UTC
Type6 masterkey operation is inprogress

Masterkey upate status information:
Client Name                Status
====================================
keychain                   INPROGRESS
```

Clear Type 6 Client State

You can use the **clear type6 client** command in XR EXEC mode to clear the Type 6 client state.

If the primary key update operation is stuck at any stage, then you can use this **clear** command to clear that state. You can track the primary key update operation using the **show type6 server** command output. If the

Master key Inprogress field in that output displays as *YES*, then you can use **show type6 masterkey update status** command (or, **show type6 clients** command, prior to Release 7.0.2) to check which client has not completed the operation. Accordingly, you can clear that particular client using the **clear** command.

Associated Commands

- **clear type6 client**
- **key config-key password-encryption**
- **password6 encryption aes**
- **show type6**

Implementing Key Chain for BGP Sessions (Type 6 Client)

For detailed information about key chains, refer the *Implementing Keychain Management* chapter.

If you enable Type 6 password encryption, plain-text keys are encrypted using Type 6 encryption. Enter plain-text key-string input in alphanumeric form. If you enable MACsec with Type 6 password encryption, the key-string input is in hexadecimal format.

Configuration

```
/* Enter the key chain details */
R1 & R2# configure
R1 & R2(config)# key chain my-test-keychain
R1 & R2(config-type6_password)# key 1
```

Enter the Type 6 encrypted format using the **key-string password6** command.

Note Using the **key-string** command, you can enter the password in clear text format or Type 6 encrypted (already encrypted password) format, as used in this scenario.

Note Enable the same key string for all the routers.

```
R1 & R2 (config-type6_password-1)# key-string password6 606745575e6565$
R1 & R2 (config-type6_password-1)# cryptographic-algorithm MD5
R1 & R2 (config-type6_password-1)# accept-lifetime 1:00:00 october 24 2005 infinite
R1 & R2 (config-type6_password-1)# send-lifetime 1:00:00 october 24 2005 infinite
R1 & R2 (config-type6_password-1)# commit
```

Verification

Verify key chain trace server information.

```
R1 & R2# show key chain trace server both

Sat Jul 20 16:44:08.768 UTC
Client file lib/kc/kc_srvr_wr
4 wrapping entries (18496 possible, 64 allocated, 0 filtered, 4 total)
Jul 20 16:43:26.342 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 *********kc_srvr process
started*********
```

```
Jul 20 16:43:26.342 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 (kc_srvr) Cerrno DLL registration
successfull
Jul 20 16:43:26.349 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 (kc_srvr) Initialised sysdb connection
Jul 20 16:43:26.612 lib/kc/kc_srvr_wr 0/RP0/CPU0 t317 (kc_srvr_type6_thread) Succesfully
registered as a type6 client
```

Verify configuration details for the key chain.

```
R1 & R2# show key chain type6_password

Sat Jul 20 17:05:12.803 UTC

Key-chain: my-test-keychain -
  Key 1 -- text "606745575e656546435a4c4a476946474342535 54f49414a4f59655a486950566"
    Cryptographic-Algorithm -- MD5
    Send lifetime --  01:00:00, 24 Oct 2005 - Always valid  [Valid now]
    Accept lifetime -- 01:00:00, 24 Oct 2005 - Always valid [Valid now]
Verify Type 6 client information.
```

Associated Commands

- **key chain**
- **key-string password6**
- **show key chain trace server both**

Creating a BGP Session (Type 6 Password Encryption Use Case)

This example provides iBGP session creation configuration. To know how to configure the complete iBGP network, refer the *BGP Configuration Guide* for the platform.

Configuration Example

```
/* Create BGP session on Router1 */
R1# configure
R1(config)# router bgp 65537
```

Ensure that you use the same key chain name for the BGP session and the Type 6 encryption (for example, *my-test-keychain* in this scenario).

```
R1 (config-bgp)# neighbor 10.1.1.11 remote-as 65537
R1 (config-bgp)# keychain my-test-keychain
R1 (config-bgp)# address-family ipv4 unicast
R1 (config-bgp)# commit
```

Repeat the above steps on Router 2 as well.

Ensure that you use the same session and keychain for all the routers (R1 and R2 in this case).

```
/* Create BGP session on Router2 */
R2 (config)# router bgp 65537
R2 (config-bgp)# neighbor 10.1.1.1 remote-as 65537
R2 (config-bgp)# keychain my-test-keychain
R2 (config-bgp)# address-family ipv4 unicast
R2 (config-bgp)# commit
```

Verification

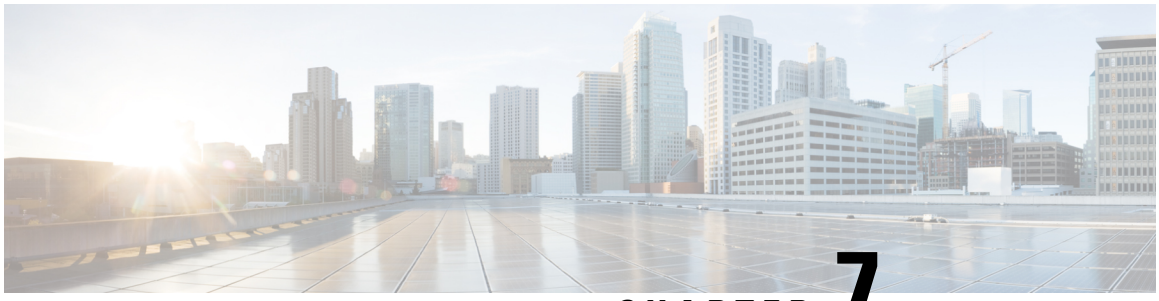
On the routers R1 and R2, verify that the BGP NBR state is in the *Established* state.

```
R1# show bgp sessions
Neighbor      VRF      Spk    AS      InQ  OutQ  NBRState     NSRState
10.1.1.11     default  0      65537   0    0     Established  None

R2# show bgp sessions
Neighbor      VRF      Spk    AS      InQ  OutQ  NBRState     NSRState
10.1.1.1     default  0      65537   0    0     Established  None
```

Associated Commands

- **session-group**
- **show BGP sessions**



CHAPTER 7

802.1X Port-Based Authentication

The IEEE 802.1X port-based authentication protects the network from unauthorized clients. It blocks all traffic to and from devices at the interface, until the Authentication server authenticates the client. After successful authentication, the port is open for traffic.

This chapter describes how to configure IEEE 802.1X port-based authentication in Cisco NCS 560 Series Routers to prevent unauthorized devices (clients) from gaining access to the network.

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Layer 2 Untagged Sub-interface configuration in IEEE 802.1X Port-based Authentication	Release 24.2.1	This feature enhances network security by extending the 802.1X port-based authentication to Layer 2 untagged sub-interfaces. It ensures that data transmission is only possible from authenticated devices, including those on interfaces without VLAN tags. Consequently, this reinforces access control policies across all devices attempting to connect to the network.

Table 17: Feature History

Release	Modification
Release 7.2.1	Support for multi-auth and multi-host modes by 802.1X to allow multiple hosts or MAC addresses on a single port was introduced.
Release 6.6.3	This feature was introduced.

- Usage guidelines and restrictions for 802.1X port-based authentication, on page 156
- IEEE 802.1X Device Roles, on page 156
- Understanding 802.1X Port-Based Authentication, on page 157
- 802.1X host-modes, on page 158
- Prerequisites for 802.1X Port-Based Authentication, on page 158
- 802.1X with Remote RADIUS Authentication, on page 159
- 802.1X with Local EAP Authentication, on page 161
- Router as 802.1X Supplicant, on page 164

• Verify 802.1X Port-Based Authentication, on page 165

Usage guidelines and restrictions for 802.1X port-based authentication

Consider these restrictions and usage guidelines when implementing 802.1X port-based authentication on the Cisco 8000 platform:

Port authentication

- 802.1X port authentication must be configured on physical ports.
- Supported modes for 802.1X port-based authentication:
  - Single-host
  - Multi-auth

VLAN sub-interfaces

- VLAN sub-interfaces must have pre-configured VLAN IDs.
- All VLAN-tagged traffic is dropped until successful 802.1X authentication of the port.
- No default VLAN assignment is provided for unauthenticated MAC addresses.
- Authenticated MAC addresses are validated at the main port, independent of VLAN assignment.
- VLAN-tagged traffic is allowed only for authenticated MAC addresses.

Untagged traffic

- Untagged EAPOL traffic is always allowed.
- All other untagged traffic is dropped until successful 802.1X authentication of the port.
- Untagged traffic is allowed only for authenticated MAC addresses.
- No default VLAN assignment is provided for untagged traffic by the port.

IEEE 802.1X Device Roles

The devices in the network have the following specific roles with IEEE 802.1X authentication:

- **Authenticator** - An entity that facilitates authentication of other entities attached to the same LAN.
- **Supplicant** - An entity at one end of a point-to-point LAN segment that seeks to be authenticated by an Authenticator attached to the other end of that link.

- **Authentication Server** - An entity that provides an authentication service to an Authenticator. Based on the credentials provided by the Supplicant, the server determines whether the Supplicant is authorized to access the services provided by the system in which the Authenticator resides.

Understanding 802.1X Port-Based Authentication

IEEE 802.1X port-based authentication is configured on Cisco 8000 series router to prevent unauthorized routers (supplicants) from gaining access to the network. An authentication server validates the supplicant that is connected to an authenticator port, before the services offered by the client or the network is made available to the supplicant.

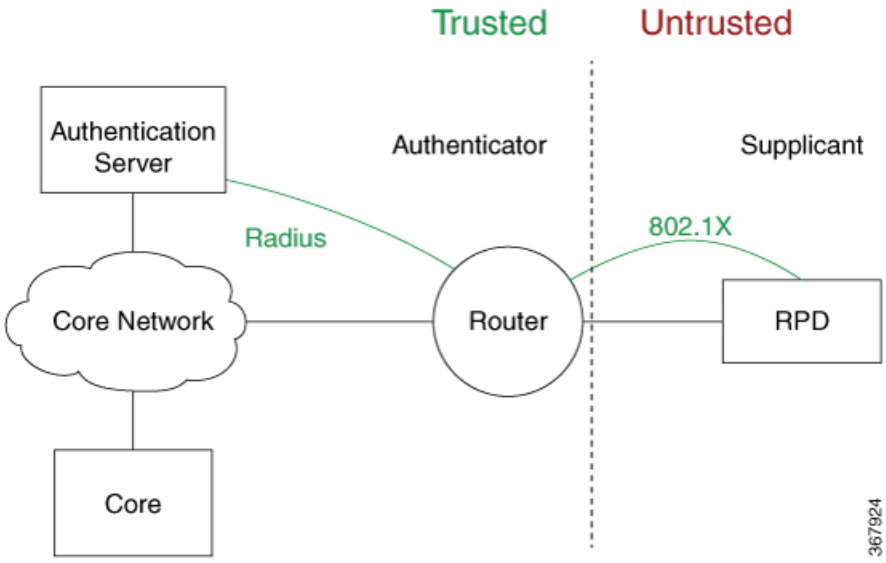
Until the supplicant is authenticated, the port is in *Unauthorized* state, and 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPOL) packets through the port. EAPoL frames can have either default EtherType of 0x888E or Cisco-defined EtherType of 0x876F. After successful authentication of the supplicant, the port transitions to *Authorized* state, and normal traffic passes through the port for the authenticated client.

Periodic reauthentication can be enabled to use either the port-configured value or from authentication server. The authentication server communicates the reauthentication-timer value in Session-Timeout attribute, with the final RADIUS Access-Accept message. On 802.1X reauthentication failure, the port is blocked and moved back to the *Unauthorized* state.

If the link state of a port changes from up to down, or if an EAPOL-logoff frame is received, the port returns to the *Unauthorized* state.

The following figure shows the topology for IEEE 802.1X port-based authentication:

Figure 4: Topology for IEEE 802.1X Port-Based Authentication



By default, the dot1x configured port is in multi-auth mode. However, this behaviour can be altered by changing the host mode under dot1x profile.

Note Port-control is enforced only on the ingress traffic.

802.1X host-modes

The following table describes the three host modes supported by 802.1X:

Table 18: 802.1X host modes

Host modes	Description
Single-host	While in this mode, the port allows a single host to be authenticated and allows only ingress traffic from the authenticated peer. A security violation is detected if more than one client is present.
Multi-auth	This is the default host mode. While in this mode, multiple hosts can independently authenticate through the same port and ingress traffic is allowed from all authenticated peers.
Multi-host	While in this mode, the first device to authenticate will open the port access so that all other hosts can use the port. These hosts need not be authenticated independently. If the authenticated host becomes unauthorized, the port will be closed.

Configure 802.1X host-modes

Use the following steps to configure 802.1X host-modes. Here, `host-mode` is introduced under the authenticator mode in dot1x profile. The default is `multi-auth` mode.

```
Router# configure terminal
Router(config)# dot1x profile {name}
Router(config-dot1x-auth)# pae {authenticator}
Router(config-dot1x-auth-auth)# host-mode
  multi-auth   multiple authentication mode
  multi-host   multiple host mode
  single-host  single host mode
```

Prerequisites for 802.1X Port-Based Authentication

Prerequisites for 802.1X port-based authentication are:

- K9sec RPM is required to enable this feature.
- Ensure that both RADIUS/EAP-server and supplicant are configured with supported EAP methods when remote authentication is used.
- If the device is used as a local EAP server, only EAP-TLS method is supported.
    - Ensure that a Certificate Authority (CA) server is configured for the network with a valid certificate.

- Ensure that the supplicant, authenticator, and CA server are synchronized using Network Time Protocol (NTP). If time is not synchronized on all these devices, certificates may not be validated.

802.1X with Remote RADIUS Authentication

Configure RADIUS Server

To configure RADIUS server pre-shared keys, obtain the pre-shared key values for the remote RADIUS server and perform this task.

Configuration Example

```
Router# configure terminal
Router(config)# radius-server host 209.165.200.225 auth-port 1646 key secret007
Router(config)# radius-server vsa attribute ignore unknown
Router(config)# commit
```

Running Configuration

```
Router# show run radius
radius-server host 209.165.200.225 auth-port 1646
  key 7 00171605165E1F565F76
radius-server vsa attribute ignore unknown
!
```

For more information, see Configure Router to RADIUS Server Communication, on page 31 and Configure RADIUS Server Groups, on page 38 in chapter *Configuring AAA Services*.

Configure 802.1X Authentication Method

You can configure 802.1X authentication method using RADIUS as the protocol. Only default AAA method is supported for 802.1X authentication.

Configuration Example

```
Router# configure terminal
Router(config)# aaa authentication dot1x default group radius
Router(config)# commit
```

Running Configuration

```
Router# show run aaa
  aaa authentication dot1x default group radius
```

Configure 802.1X Authenticator Profile

Configure 802.1X profile on an authenticator.

```
RP/0/RP0/CPU0:router(config)# dot1x profile <auth>
RP/0/RP0/CPU0:router(config-dot1x-auth)# pae authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth)# authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# timer reauth-time 3600
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# host-mode { multi-auth | multi-host |
single-host }
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# commit
```

Running Configuration

The following is a sample output of `show run dot1x` command.

```
RP/0/RSP0/CPU0:router# show run dot1x profile auth
dot1x profile auth
pae authenticator
authenticator
  timer reauth-time 3600
  host-mode multi-auth
!
```

Configure 8021X Profile on Interface

You can attach one of the 802.1X profiles on an interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface <interface-name>
RP/0/RSP0/CPU0:router(config-if)# dot1x profile <profile-name>
RP/0/RSP0/CPU0:router(config-if)# commit
```

Example Configuration

This example provides the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE 0/3/0/0
RP/0/RSP0/CPU0:router(config-if)# dot1x profile auth
RP/0/RSP0/CPU0:router(config-if)# commit
```

This example verifies the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
  dot1x profile auth
```

This example provides the configuration to allow tagged traffic with VLAN IDs 1 & 2:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.1
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.1.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 1
!
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.2
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.2.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)#encapsulation dot1q 2
!
```

This example provides the configuration to allow untagged Layer 2 sub-interface traffic:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config-subif)# interface HundredGigE0/3/0/0.1 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation untagged
!
```

802.1X with Local EAP Authentication

In local EAP authentication, the EAP-server is co-located with the authenticator locally on the router. This feature enables the router to authenticate 802.1X clients with EAP-TLS method using TLS Version 1.2. It provides EAP-TLS based mutual authentication, where a Master Session Key (MSK) is generated on successful authentication.

Generate RSA Key Pair

RSA key pairs are used to sign and encrypt key management messages. This is required before you can obtain a certificate for the node.

```
RP/0/RSP0/CPU0:router#crypto key generate rsa < keypair-label >
```

Running Configuration

The following is a sample output of **show crypto key mypubkey rsa** command.

```
RP/0/RSP0/CPU0:router# show crypto key mypubkey rsa
Key label:  rsa_tp
Type : RSA General purpose
Size : 2048
Data :
30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
00BAA4F5 19C1C41A 4A195B31 6722B853 5271EECA B884CC19 CE75FB23 19DC0346
2F90F9B2 CBCB9BA3 4E4DDD46 2C21F555 4C642E3A 98FE0A2F 587D79F5 1D5B898F
893CEC38 B7C8CB03 48D0AEA1 D554DF2B BA751489 3099A890 1A910D25 7DA78F99
E29526FE 6F84C147 4F872715 D3BDE515 FACB28E8 6375BB38 1F3AFDA8 853C6E57
8BDA1800 7DDADFE3 32ABAB4C 3D078342 36E79F05 CAFCE764 26274F41 25F7BC70
04ABEDFE 96A183EE 23A3D099 2D5741C5 F81747FB 1ED5F672 5449B7AE 8D2E9224
CF12E1CA 9E2373C4 41BF29FA A9DDD930 5A3A2FDE FD1DADE1 2548DEDB 05FC2176
7D5DB337 B1563CA3 A94DF081 5B294D1A A9B70A56 CA5CF7B2 A779F27A 3EE4F568
F1020301 0001
```

For more information, see Generate RSA Key Pair, on page 107 in chapter *Implementing Certification Authority Interoperability*.

Configure Trustpoint

Trustpoints let you manage and track CAs and certificates. A trustpoint includes the identity of the CA, CA-specific configuration parameters, and an association with one, enrolled identity certificate. After you have defined a trustpoint, you can reference it by name in commands requiring that you specify a CA.

```
RP/0/RSP0/CPU0:router# configure terminal
RP/0/RSP0/CPU0:router(config)# crypto ca trustpoint <tp_name>
RP/0/RSP0/CPU0:router(config-trustp)# enrollment url <ca-url>
RP/0/RSP0/CPU0:router(config-trustp)# subject-name <x.500-name>
RP/0/RSP0/CPU0:router(config-trustp)# rsakeypair <keypair-label>
RP/0/RSP0/CPU0:router(config-trustp)# crl optional
RP/0/RSP0/CPU0:router(config-trustp)# commit
```

Running Configuration

The following is a sample output of **show run crypto ca trustpoint** *tp_name* command.

```
crypto ca trustpoint tp
  crl optional
```

```
subject-name CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
enrollment url http://20.30.40.50
rsakeypair rsa_tp
 !
```

For more information, see Declare Certification Authority and Configure Trusted Point, on page 109 in chapter *Implementing Certification Authority Interoperability*.

Configure Domain Name

You can configure a domain name, which is required for certificate enrolment.

```
RP/0/RSP0/CPU0:router# domain name ca.cisco.com
```

Running Configuration

The following is a sample output of **show run domain name** command.

```
RP/0/1/CPU0:router# show run domain name
Thu Mar 29 16:10:42.533 IST
domain name cisco.com
```

Certificate Configurations

Certificate enrolment involves the following two steps:

1. Obtain CA certificate for the given trust point, using the **crypto ca authenticate** *tp_name* command.
2. Enroll the device certificate with CA, using the **crypto ca enroll** *tp_name* command.

```
RP/0/RSP0/CPU0:router# crypto ca authenticate <tp_name>
RP/0/RSP0/CPU0:router# crypto ca enroll <tp_name>
```

Running Configuration

The following is a sample output of the **show crypto ca certificates** command.

```
RP/0/RSP0/CPU0:router# show crypto ca certificates
Trustpoint : tp
==================================================
CA certificate
Serial Number        : E0:18:F3:E4:53:17:3E:28
Subject              : subject-name CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Issued By            : subject-name CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Validity Start       : 08:17:32 UTC Fri Jun 24 2016
Validity End         : 08:17:32 UTC Mon Jun 22 2026
SHA1 Fingerprint     : 894ABBFAA3B08E5B7D9E470ECFBBC04576B569F2
Router certificate
Key usage            : General Purpose
Status               : Available
Serial Number        : 03:18
Subject              :
serialNumber=cf302761,unstructuredAddress=20.30.40.50,unstructuredName=asr9k,
C=US,ST=NY,L=Newyork,O=Govt,OU=BU,CN=asr9k
Issued By            : CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Validity Start       : 13:04:52 UTC Fri Feb 23 2018
Validity End         : 13:04:52 UTC Sat Feb 23 2019
SHA1 Fingerprint     :33B50A59C76CCD87D3D0F0271CD5C81F4A1EE9E1
Associated Trustpoint: tp
```

For more information, see Declare Certification Authority and Configure Trusted Point, on page 109 in chapter *Implementing Certification Authority Interoperability*.

Configure EAP Profile

You can configure multiple EAP profiles.

```
RP/0/RSP0/CPU0:router# configure terminal
RP/0/RSP0/CPU0:router(config)# eap profile <name>
RP/0/RSP0/CPU0:router(config-eap)# identity <user-name>
RP/0/RSP0/CPU0:router(config-eap)# method tls pki-trustpoint <trustpoint-name>
RP/0/RSP0/CPU0:router(config-eap)# commit
```



Note To allow EAP-TLS authentication with peer devices or EAP-server running on TLS 1.0, configure `allow-eap-tls-v1.0` under EAP profile.

Running Configuration

The following is sample output of **show run eap** command.

```
RP/0/RSP0/CPU0:router# show run eap profile <local eap>
eap profile local_eap
method tls
  pki-trustpoint tp
!
identity CE1
```

Configure 802.1X Authenticator Profile

You can configure 802.1X profile on an authenticator.

```
RP/0/RP0/CPU0:router(config)# dot1x profile local_auth
RP/0/RP0/CPU0:router(config-dot1x-auth)# pae authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth)# authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# eap profile <local_eap>
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# host-mode {multi-auth | multi-host |
single-host}
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# timer reauth-time 3600
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# commit
```

Running Configuration

The following is a sample output of `show run dot1x` command.

```
RP/0/RSP0/CPU0:router# show run dot1x profile local_auth

dot1x profile local_auth
pae authenticator
  authenticator
     eap profile local_eap
     host-mode multi-host
     timer reauth-time 3600
```

Configure 802.1X Profile on Interface

You can attach one of the 802.1X profiles on an interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface <interface-name>
RP/0/RSP0/CPU0:router(config-if)# dot1x profile <profile-name>
RP/0/RSP0/CPU0:router(config-if)# commit
```

Example Configuration

This example provides the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE 0/3/0/0
RP/0/RSP0/CPU0:router(config-if)# dot1x profile auth
RP/0/RSP0/CPU0:router(config-if)# commit
```

This example verifies the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
  dot1x profile auth
```

This example provides the configuration to allow tagged traffic with VLAN IDs 1 & 2:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.1
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.1.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 1
!
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.2
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.2.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)#encapsulation dot1q 2
!
```

This example provides the configuration to allow untagged Layer 2 sub-interface traffic:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config-subif)# interface HundredGigE0/3/0/0.1 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation untagged
!
```

Router as 802.1X Supplicant

To configure the router as 802.1X supplicant, make sure that the following configurations are enabled:

- RSA Key Pair: Generate RSA Key Pair, on page 161
- Trust point: Configure Trustpoint, on page 161
- Domain name: Configure Domain Name, on page 162
- Certificates: Certificate Configurations, on page 162
- EAP profile: Configure EAP Profile, on page 163

Configure 802.1X Supplicant Profile

You can configure 802.1X profile on a supplicant.

```
RP/0/RP0/CPU0:router(config)# dot1x profile supp
RP/0/RP0/CPU0:router(config-dot1x-supp)# pae supplicant
RP/0/RP0/CPU0:router(config-dot1x-supp)# supplicant
RP/0/RP0/CPU0:router(config-dot1x-supp-supp)# eap profile eap_supp
RP/0/RP0/CPU0:router(config-dot1x-supp-supp)# commit
```

Running Configuration

The following is a sample output of `show run dot1x` command.

```
RP/0/RSP0/CPU0:router# show run dot1x profile supp
dot1x profile supp
pae supplicant
supplicant
eap profile eap_supp
!
```

Configure 802.1X Profile on Interface

You can attach one of the 802.1X profiles on an interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface <interface-name>
RP/0/RSP0/CPU0:router(config-if)# dot1x profile <profile-name>
RP/0/RSP0/CPU0:router(config-if)# commit
```

Example Configuration

```
Router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
  dot1x profile supp
```

Verify 802.1X Port-Based Authentication

The 802.1X authentication can be verified using the following:

- Show command outputs
- Syslog messages

Show Command Outputs

The **show dot1x interface** command verifies whether the 802.1X port-based authentication is successful or not. If the authentication is successful, the traffic is allowed on the configured interface.

```
Router# show dot1x interface HundredGigE 0/0/1/0 detail

Dot1x info for HundredGigE 0/0/1/0
-----------------------------------------------------------------
Interface short name      : Hu 0/0/1/0
Interface handle          : 0x4080
Interface MAC             : 021a.9eeb.6a59
Ethertype                 : 888E
PAE                       : Authenticator
Dot1x Port Status         : AUTHORIZED
Dot1x Profile             : test_prof
L2 Transport              : FALSE
Authenticator:
   Port Control           : Enabled
```

```
   Config Dependency      : Resolved
   Eap profile            : None
   ReAuth                 : Disabled
Client List:
      Supplicant          : 027e.15f2.cae7
 Programming Status       : Add Success
      Auth SM State       : Authenticated
      Auth Bend SM State  : Idle
      Last authen time    : 2018 Dec 11 17:00:30.912
      Last authen server  : 10.77.132.66
      Time to next reauth : 0 day(s), 00:51:39
MKA Interface:
   Dot1x Tie Break Role   : NA (Only applicable for PAE role both)
   EAP Based Macsec       : Disabled
   MKA Start time         : NA
   MKA Stop time          : NA
   MKA Response time      : NA


Router#show dot1x
Mon Jun 15 18:30:49.327 IST

NODE: node0_RP0_CPU0

Dot1x info for TenGigE0/11/0/1
------------------------------------------------------------
PAE                       : Authenticator
Dot1x Port Status         : AUTHORIZED (2/2)
Dot1x Profile             : auth
Authenticator:
   Host Mode              : Multi-Auth
   Port Control           : Enabled
   Config Dependency      : Resolved
   Eap profile            : Not Configured
   ReAuth                 : Enabled, 1 day(s), 00:00:00
   Client List:
      Supplicant          : 008a.96a4.b028
      Port Status         : Authorized
      Programming Status  : Add Success
      Auth SM State       : Authenticated
      Auth Bend SM State  : Idle
      Last authen time    : 2020 Jun 15 18:30:42.659
      Last authen server  : 10.105.236.94
      Time to next reauth : 0 day(s), 23:59:52

      Supplicant          : 008a.96a4.c830
      Port Status         : Authorized
      Programming Status  : Add Success
      Auth SM State       : Authenticated
      Auth Bend SM State  : Idle
      Last authen time    : 2020 Jun 15 18:30:42.654
      Last authen server  : 10.105.236.94
      Time to next reauth : 0 day(s), 23:59:52
```

Syslog Messages

Syslogs on Authenticator

When 802.1x configuration is applied on an interface, the port becomes 802.1X controlled, and the following syslog message is displayed:

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled

%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Single-Host
 mode

%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Multi-Host
 mode

%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Multi-Auth
 mode
```

When there is a host-mode violation, the following syslog messages are displayed:

```
%L2-DOT1X-3-HOST_MODE_VIOLATION: Hu0/0/1/0 : multiple clients detected in Single-Host mode,
 dropping supplicant (008a.9686.0058) request

%L2-DOT1X-3-HOST_MODE_VIOLATION: Hu0/0/1/0 : multiple clients detected in Multi-Host mode,
 dropping supplicant (008a.9686.0058) request
```

After successful authentication of supplicant, the following syslog messages are displayed:

```
%L2-DOT1X-5-AUTH_SUCCESS : Hu0/0/1/0 : Authentication successful for client 027E.15F2.CAE7


%L2-DOT1X-5-PORT_CONTROL_ADD_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Enabled For Client
027E.15F2.CAE7
```

When 802.1X port-based configuration is removed from an interface, the following syslog message is displayed:

```
%L2-DOT1X-5-PORT_CONTROL_DISABLE_SUCCESS : Hu0/0/1/0  : Port Control Disabled
```

When authentication fails, the following syslog messages are displayed:

```
%L2-DOT1X-5-AUTH_FAIL : Hu0/0/1/0 : Authentication fail for client 027E.15F2.CAE7

%L2-DOT1X-5-PORT_CONTROL_REMOVE_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Disabled For Client
 027E.15F2.CAE7
```

When authentication server is unreachable, the following syslog message is displayed:

```
%L2-DOT1X-5-AAA_UNREACHABLE : Hu0/0/1/0 : AAA server unreachable for client 027E.15F2.CAE7
  , Retrying Authentication
```

When authentication method is not configured, the following syslog message is displayed:

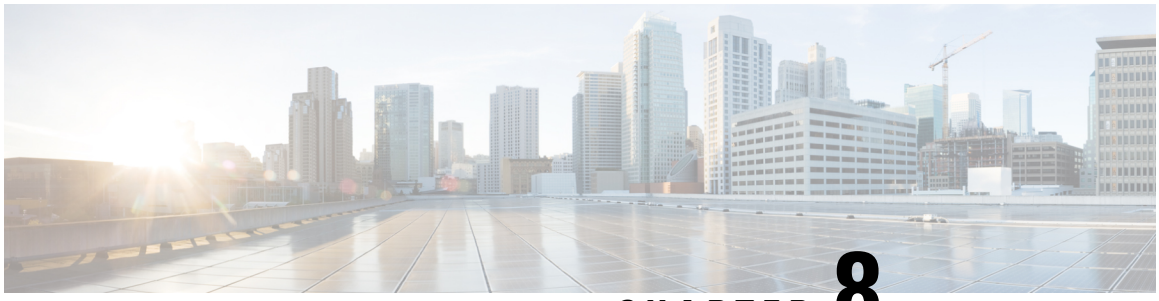
```
%L2-DOT1X-4-NO_AUTHENTICATION_METHOD : Hu0/0/1/0 : No authentication method configured
```

Syslogs on Supplicant

```
%L2-DOT1X-5-SUPP_SUCCESS : Hu0/0/1/0 : Authentication successful with authenticator
008a.96a4.b050

%L2-DOT1X-5-SUPP_FAIL : Hu0/0/1/0 : Authentication successful with authenticator
0000.0000.0000.0000

%L2-DOT1X-5-SUPP_FAIL : Hu0/0/1/0 : Authentication successful with authenticator
008a.96a4.b028
```



CHAPTER 8

Implementing MAC Authentication Bypass

This chapter describes the implementation of MAC Authentication Bypass (MAB).

IEEE 802.1X authentication configuration on the router helps to prevent unauthorized end devices from gaining access to the network. However, not all end devices support 802.1X. Hence, we introduce port controlling functionality on these routers using MAC authentication bypass (MAB)—a feature that grants network access to devices based on their MAC addresses, regardless of their 802.1X capability or credentials.

For details of commands related to MAB, see the *802.1X and Port Control Commands* chapter in the *System Security Command Reference for Cisco NCS 5500 Series Routers and Cisco NCS 540 and NCS 560 Series Routers*.

- MAC Authentication Bypass, on page 170

MAC Authentication Bypass

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
MAC Authentication Bypass	Release 24.3.1	Based on the MAC address of the end device or the client connected to the router port, this feature enables port control functionality for your router. This functionality provides controlled access to network services for end devices that do not support other authentication methods such as IEEE 802.1X port-based authentication. The feature introduces these changes: **CLI:** • New **mab** option for the **dot1x profile** command • New **mab-retry-time** option for the **authenticator** command • **clear mab** • **show mab**

MAC authentication bypass (MAB) is a port control feature in which the router (authenticator) uses the MAC address of the end device or the client (also called as supplicant) as an authenticating parameter to provide network access.

802.1X (Dot1x) is one of the most widely used port-based authentication method to allow controlled access to the end devices connected to the port. However, not all clients support 802.1X. We would still need to allow them into the network even without 802.1X authentication. The MAB feature intends to provide this controlled access to such devices based on their MAC addresses.

Restrictions for MAC Authentication Bypass

These restrictions apply to the MAB feature:

- With MAB, you can perform user authentication using a remote AAA server only; not using the local AAA server on the router.
- MAB feature works only as a standalone feature; not as a fallback mechanism for any other type of authentication failures.

- MAB supports only a single end device on each port.

  Hence, you must configure the authenticator (the router) to be in **single-host** mode.

Authentication Failure Scenarios of MAB

This table lists the various authentication failure scenarios and the expected behavior of MAB feature.

Table 20: Authentication Failure Scenarios and Expected Feature Behavior of MAB

If...	And...	Then…
the RADIUS server rejects the authentication request	—	the router • deletes the client programming on the port, if that client was already authenticated • retries the authentication process twice with the RADIUS server at an interval (configurable using the **authenticator timer mab-retry-time** command) of 60 seconds, by default • clears the client session and its programming on the port (if the server still does not authorize the client), and • puts the port back in MAC learning mode to relearn a new MAC address.
the client is unauthenticated	authentication does not happen after the retries	the router • deletes the client context, and • puts the port back in MAC learning mode to relearn a new MAC address.
the RADIUS server is not reachable during authentication process	**server dead action auth-retry** command is configured	the router • retains the programming of the client that was already authenticated • retries the authentication process with the RADIUS server at an interval (configurable using the **authenticator timer mab-retry-time** command) of 60 seconds until the server becomes available • does not attempt to learn any new MAC address on the port, and • the router puts the port back in MAC learning mode to relearn a new MAC address. To clear the client session and its programming on the router, use the **clear mab session** command.

If...	And...	Then...
the RADIUS server is not reachable during authentication process	**server dead action auth-retry** command is not configured	the router • deletes the programming of the client that was already authenticated and retries authentication • automatically clears the client session, if the client is still not authenticated, and • puts the port back in MAC learning mode to relearn a new MAC address.

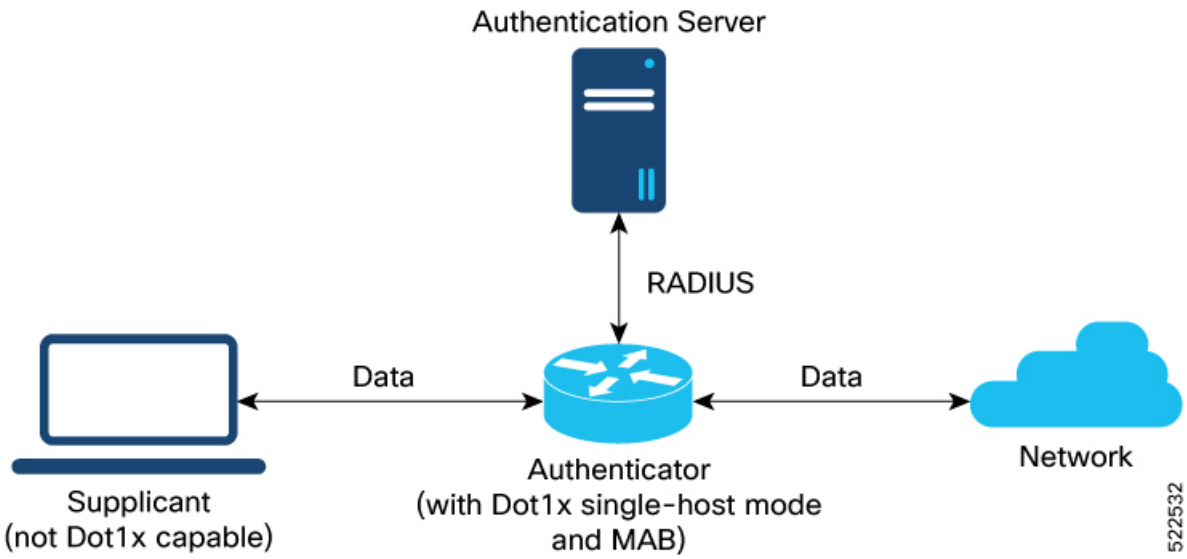
How MAC Authentication Bypass Works

Summary

These are the key components of MAC authentication bypass:

- Supplicant: The client or end device without dot1x support.
- Authenticator: The router that tries to authenticate the host device running the supplicant with the authentication server.
- Authentication Server: The server that provides the authenticator the RADIUS reply (**Access-Accept** or **Access-Reject** message), which allows or denies network access to the end device.

Workflow



These stages describe how MAC authentication bypass process works.

1. The router receives an incoming data packet from the client that is connected to the router port.
2. The router

a. learns the source MAC address, and

b. sends it to the external RADIUS server (authentication server) for authentication.

   The RADIUS authentication server maintains a database of MAC addresses for devices that require access to the network.

3. Based on the authentication result, the router allows or drops the data packets from that client.

If the RADIUS server…	Then…	And…
returns a success (**Access-Accept**) message	• it indicates that the MAC address is authenticated • the client is authorized to send traffic through that port	the router allows the traffic from the client to be forwarded to the network.
returns a failure (**Access-Reject**) message	it indicates that the MAC address is unauthenticated	the router drops further data packets from that client.

Result

Thus, the MAB feature brings in port control functionality for Cisco IOS XR routers and provides end devices a controlled access to network services.

Configure MAC Authentication Bypass

Before you begin

- Configure the remote RADIUS server using the **radius-server** command, and authentication method with the RADIUS server using the **aaa authentication dot1x** command in .
- Configure the 802.1X profile (using the **dot1x profile** command in XR Config mode)
- Configure the authenticator (using the **authenticator** command in dot1x profile configuration sub mode) by specifying these parameters:
  - Re-authentication time—using **reauth-time** option
  - Host mode—using **single-host** option
  - Retry action for server-unreachable scenarios—using **auth-retry** or **auth-fail** option

See the *MACSec Using EAP-TLS Authentication* chapter for these configuration details.

See *Running Configuration* section for examples.

Follow these steps to configure MAC authentication bypass feature.

Procedure

Step 1 Enable MAB on the dot1x profile.

Example:

```
Router#configure
Router(config)#dot1x profile test_mab
Router(dot1xx-test_mab)#mab
Router(dot1xx-test_mab)#commit
```

Step 2 Configure the authenticator retry time for MAB clients.

Example:

```
Router#configure
Router(config)#dot1x profile test_mab
Router(dot1xx-test_mab)#authenticator
Router(dot1xx-test_mab-auth)#timer mab-retry-time 60
Router(dot1xx-test_mab-auth)#commit
```

Step 3 Attach the dot1x profile to the corresponding interface or port on the router.

Example:

```
Router(config)#interface GigabitEthernet0/0/0/0
Router(config-intf)#dot1x profile test_mab
Router(config-intf)#commit
```

Step 4 Verify the running configuration on the router.

Example:

```
Router# show running-configuration

!
radius-server host <ip-address> auth-port <auth-port-num> acct-port <acct-port-num>
 key 7 <key>
!
aaa authentication dot1x default group radius
interface GigabitEthernet0/0/0/0
 dot1x profile test_mab
!

dot1x profile test_mab
 mab
 authenticator
  timer reauth-time 60
  timer mab-retry-time 60
  host-mode single-host
  server dead action auth-retry
 !
!
end
```

Verify MAC Authentication Bypass

Follow these steps to verify MAC authentication bypass feature.

Procedure

Step 1 Check the MAB summary.

Example:

```
Router#show mab summary
Fri Apr  1 16:37:32.340 IST

NODE: node0_0_CPU0
==============================================================================
   Interface-Name        Client          Status
==============================================================================
     Gi0/0/0/0         1122.3344.5566   Authorized
Router#
```

The *Status* field shows as **Authorized**.

Step 2 Verify the detailed status of MAB.

Example:

```
Router#show mab detail
Fri Apr  1 16:37:37.140 IST

NODE: node0_0_CPU0

MAB info for GigabitEthernet0/0/0/0
------------------------------------------------------------------
InterfaceName       : Gi0/0/0/0
InterfaceHandle     : 0x00000060
HostMode            : single-host
PortControl         : Enabled
PuntState           : Stop Success
PuntSummary         : Punt disabled
Client:
  MAC Address       : 1122.3344.5566
  Status            : Authorized
  SM State          : Terminate
  ReauthTimeout     : 60s, Remaining 0 day(s), 00:00:46
  RetryTimeout      : 60s, timer not started yet
  AuthMethod        : PAP (remote)
  LastAuthTime      : 2022 Apr 01 16:37:23.634
  ProgrammingStatus : Add Success
Router#
```

The *PortControl* field shows as **Enabled**.

Step 3 Verify the MAB interface summary.

Example:

```
Router#show mab interface gigabitEthernet 0/0/0/0
Fri Apr  1 16:38:27.715 IST
==============================================================================
```

```
   Interface-Name        Client          Status
========================================================================
     Gi0/0/0/0         1122.3344.5566   Authorized
```

The *Status* field shows as **Authorized**.

Step 4 Verify the MAB interface details.

Example:

```
Router#show mab interface gigabitEthernet 0/0/0/0 detail
Fri Apr  1 16:38:31.543 IST
MAB info for GigabitEthernet0/0/0/0
------------------------------------------------------------------
InterfaceName       : Gi0/0/0/0
InterfaceHandle     : 0x00000060
HostMode            : single-host
PortControl         : Enabled
PuntState           : Stop Success
PuntSummary         : Punt disabled
Client:
  MAC Address       : 1122.3344.5566
  Status            : Authorized
  SM State          : Terminate
  ReauthTimeout     : 60s, Remaining 0 day(s), 00:00:51
  RetryTimeout      : 60s, timer not started yet
  AuthMethod        : PAP (remote)
  LastAuthTime      : 2022 Apr 01 16:38:23.640
  ProgrammingStatus : Add Success
Router#
```

The *PortControl* field shows as **Enabled**.

Step 5 Verify the MAB interface statistics.

Example:

```
Router#show mab statistics interface gigabitEthernet 0/0/0/0
Fri Apr  1 16:41:23.011 IST
InterfaceName         : GigabitEthernet0/0/0/0
------------------------------------------------------------------
MAC Learning:
  RxTotal             : 0
  RxNoSrcMac          : 0
  RxNoIdb             : 0
Port Control:
  EnableSuccess       : 1
  EnableFail          : 0
  UpdateSuccess       : 0
  UpdateFail          : 0
  PuntStartSuccess    : 0
  PuntStartFail       : 0
  PuntStopSuccess     : 1
  PuntStopFail        : 0
  AddClientSuccess    : 1
  AddClientFail       : 0
  RemoveClientSuccess : 0
  RemoveClientFail    : 0
Client                :
  MAC Address         : 1122.3344.5566
  Authentication:
    Success           : 1406
    Fail              : 0
```

```
    Timeout           : 0
    AAA Unreachable   : 0
Router#
```

The *EnableSuccess* field under *Port Control* shows as **1**.

System Logs for MAC Authentication Bypass

The router displays these system logs on the console in various MAB scenarios:

- When you apply dot1x profile on the port, with MAB feature enabled

  Success case:

  ```
  %L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with
  Single-Host mode
  ```

  Failure case:

  ```
  %L2-DOT1X-5-PORT_CONTROL_ENABLE_FAILURE : Hu0/0/1/0 : Failed to enable port-control
  ```

- When you remove dot1x profile from the interface

  Success case:

  ```
  %L2-DOT1X-5-PORT_CONTROL_DISABLE_SUCCESS : Hu0/0/1/0 : Port Control Disabled
  ```

  Failure case:

  ```
  %L2-DOT1X-5-PORT_CONTROL_DISABLE_FAILURE : Hu0/0/1/0 : Failed to disable port-control
  ```

- As part of MAB client authentication process

  Success case:

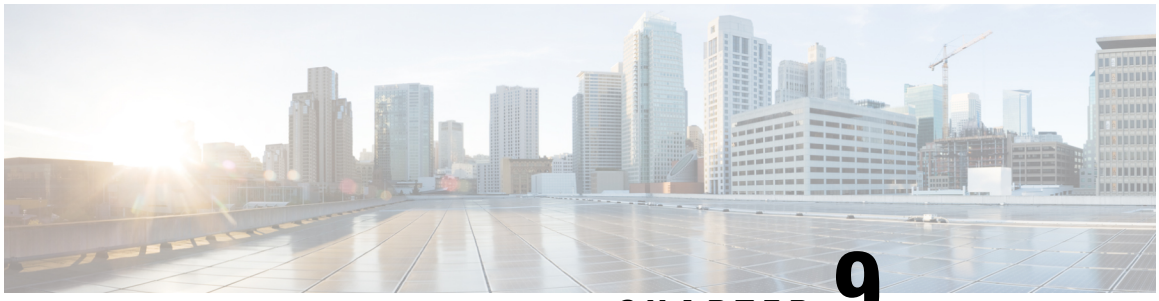
  ```
  %L2-DOT1X-5-MAB_AUTH_SUCCESS : Hu0/0/1/0 : Authentication successful for client
  <mac-address>
  %L2-DOT1X-5-PORT_CONTROL_ADD_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Enabled For Client
   <mac-address>
  ```

  Failure case:

  ```
  %L2-DOT1X-5-MAB_AUTH_FAIL    : Hu0/0/1/0 : Authentication failed for client <mac-address>
  %L2-DOT1X-5-PORT_CONTROL_REMOVE_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Disabled For
  Client <mac-address>
  ```

- When the authentication server is unreachable

  ```
  %L2-DOT1X-5-MAB_AAA_UNREACHABLE : Hu0/0/1/0 : AAA server unreachable for client
  027E.15F2.CAE7, Retrying Authentication
  ```



CHAPTER 9

Understanding URPF

It has become a commonplace practice for hackers planning a DoS attack to use forged IP addresses (the practice is known as IP address spoofing) and constantly change the source IP address to avoid detection by service providers.

Unicast Reverse Path Forwarding (URPF) is a mechanism for validating the source IP address of packets received on a router. A router configured with URPF performs a reverse path lookup in the FIB table to validate the presence of the source IP address. If the source IP address is listed in the table, then it indicates that the source is reachable and valid. If source IP address cannot be located in the FIB table, the packet is treated as malicious by the router and discarded.

The router supports the use of URPF in loose mode.URPF loose mode is enabled when the router is configured to validate only the prefix of the source IP address in the FIB and not the interface used by the packet to reach the router. By configuring loose mode, legitimate traffic that uses an alternate interface to reach the router is not mistaken to be malicious. URPF loose mode is very useful in multi-homed provider edge networks.

- Configuring uRPF Loose Mode, on page 179
- uRPF strict mode with external TCAM, on page 180
- Configure uRPF strict mode , on page 181

Configuring uRPF Loose Mode

This section explains how you can configure uRPF loose mode on the router for both IPv4 and IPv6 networks.

Before You Begin

Before you can configure uRPF loose mode on a router, you must disable the default scale on the Cisco NCS 5500 series line card, as described in this section.

For all types of line cards without TCAM:

```
RP/0/RP0/CPU0:router(config)# hw-module fib ipv4 scale host-optimized-disable

RP/0/RP0/CPU0:router(config)# commit
RP/0/RP0/CPU0:router(config)# end
RP/0/RP0/CPU0:router# reload location all
Proceed with reload? [confirm]
```

Configuration

Use the following configuration to configure uRPF loose mode on the router.

Note You must configure both IPv4 and IPv6 commands (as described in this section) for uRPF to work.

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether1
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.0.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)# ipv4 verify unicast source reachable-via any
RP/0/RP0/CPU0:router(config-if)# ipv6 address 2001::1/64
RP/0/RP0/CPU0:router(config-if)# ipv6 verify unicast source reachable-via any
RP/0/RP0/CPU0:router(config-if)# commit
```

Running Configuration

Confirm your configuration as shown:

```
RP/0/RP0/CPU0:router(config-if)# show running-config
Thu Jul 27 14:40:38.167 IST
...
!
interface Bundle-Ether1
 ipv4 address 10.0.0.1 255.255.255.0
 ipv4 verify unicast source reachable-via any
ipv6 address 2001::1/64
 ipv6 verify unicast source reachable-via any
!
```

You have successfully configured uRPF loose mode on the router.

uRPF strict mode with external TCAM

A uRPF strict mode is a network security mechanism that

- verifies that incoming packets are received on the interface used to reach the source address,
- drops packets when the source does not match the expected return-path interface, and
- prevents IP spoofing and enforces routing symmetry.

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
uRPF strict mode	Release 26.2.1	uRPF strict mode increases network security by blocking IP spoofing and dropping packets whose source IP does not match the expected return path interface. It checks that incoming traffic uses the same interface that the router would use to reach the source IP address. uRPF strict modes supports up to 16 Equal-Cost Multipath (ECMP) paths. This release extends uRPF strict mode to devices with external TCAM.

Restrictions for uRPF strict mode with external TCAM

- You cannot enable uRPF strict mode if either **hw-module profile segment-routing srv6** or **hw-module profile qos ipv6 short** is enabled.
- Strict uRPF mode is not supported on NCS 5500 line cards without external TCAM.
- S-RTBH (Source-based Remotely Triggered Black Hole) feature is not supported with strict mode uRPF mode on NCS5500 Series routers with external TCAM.
- Strict uRPF is supported only for IPv4 and IPv6 address family routes. If MPLS LDP is configured, you must also configure LDP label filtering to prevent the allocation of local labels for access IPv4 or IPv6 prefixes.
- On NCS 5700 line cards, strict uRPF is only supported with external TCAM.

Guidelines for uRPF strict mode with external TCAM

These guidelines are only applicable to NCS 5500 fixed port routers and NCS 5500 modular port routers.

- For routers with NL (Network Logic) TCAM you need to configure these commands.
  - **hw-module tcam fib ipv4 scaledisable**
  - **hw-module profile tcam fib ipv6 unicast percent** <*percentage*>
- You don't need to configure these commands for OP (Operations) TCAM.
  - **hw-module tcam fib ipv4 scaledisable**
  - **hw-module profile tcam fib ipv6 unicast percent** <*percentage*>
- uRPF strict mode is supported by default on line cards having external TCAM, if these hw-module commands are not configured:
  - **hw-module profile qos ipv6 short**
  - **hw-module profile segment-routing srv6**
- If a route has more than 16 strict uRPF-enabled paths, you get these syslog messages mentioning the invalid configuration, its impact, and recovery steps.

```
LC/0/0/CPU0:Oct 8 18:10:42.867 IST: fib_mgr[305]:
%PLATFORM-PLAT_FIB-3-SuRPF_MAX_ECMP_REACH_ERR : ECMP paths with suRPF enabled are 18
for some prefixes. Supported suRPF enabled ECMP paths for a given prefix are 16 on this
platform. This will lead to complete traffic drop for affected prefixes. Please reduce
suRPF config to supported ECMP paths scale to recover
```

Configure uRPF strict mode

Use this procedure to configure uRPF strict mode on a router.

uRPF strict mode discards packets if the device cannot verify that the source address of each packet is reachable through the receiving interface. uRPF helps prevent attacks that rely on spoofing source addresses.

Before you begin

- Configure both IPv4 and IPv6 commands as described in Configuring uRPF Loose Mode, on page 179.

Procedure

Step 1 Enter interface configuration mode for the required interface.

Example:

```
Router(config)#interface GigabitEthernet 0/1/0/0
```

Step 2 Assign the IPv4 address to the interface.

Example:

```
Router(config-if)#ipv4 address 10.0.0.1 255.255.255.0
```

Step 3 Enable uRPF strict mode for IPv4.

Example:

```
Router(config-if)#ipv4 verify unicast source reachable-via rx
```

Step 4 Assign the IPv6 address to the interface.

Example:

```
Router(config-if)#ipv6 address  2001:DB8::1/64
```

Step 5 Enable uRPF strict mode for IPv6.

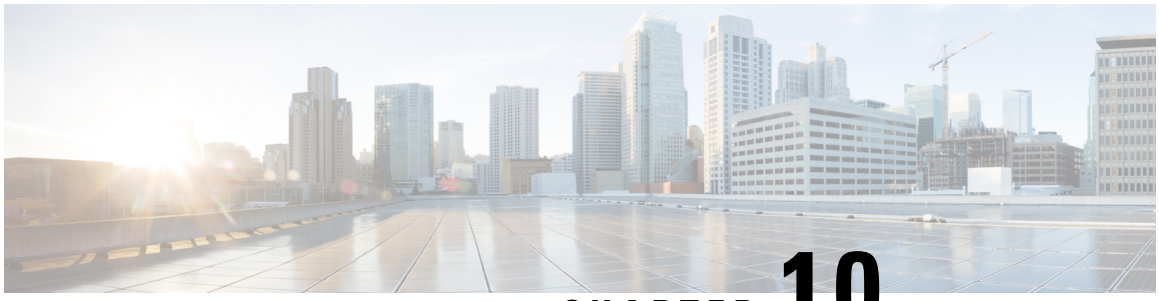
Example:

```
Router(config-if)#ipv6 verify unicast source reachable-via rx
Router(config-if)#commit
```

Step 6 Verify your configuration.

Example:

```
Router#show running-config
...
!
interface GigabitEthernet 0/1/0/0
 ipv4 address 10.0.0.1 255.255.255.0
 ipv4 verify unicast source reachable-via rx
ipv6 address  2001:DB8::1/64
 ipv6 verify unicast source reachable-via rx
!
```



CHAPTER 10

Implementing Management Plane Protection

The Management Plane Protection (MPP) feature provides the capability to restrict the interfaces on which network management packets are allowed to enter a device. The MPP feature allows a network operator to designate one or more router interfaces as management interfaces.

The MPP protection feature, as well as all the management protocols under MPP, are disabled by default. When you configure an interface as either out-of-band or inband, it automatically enables MPP. Consequently, this enablement extends to all the protocols under MPP. If MPP is disabled and a protocol is activated, all interfaces can pass traffic.

When MPP is enabled with an activated protocol, the only default management interfaces allowing management traffic are the route processor (RP) and standby route processor (SRP) Ethernet interfaces. You must manually configure any other interface for which you want to enable MPP as a management interface.

Afterwards, only the default management interfaces and those you have previously configured as MPP interfaces accept network management packets destined for the device. All other interfaces drop such packets. Logical interfaces (or any other interfaces not present on the data plane) filter packets based on the ingress physical interface.

- Benefits of Management Plane Protection, on page 183
- Restrictions for Implementing Management Plane Protection, on page 184
- Configure Device for Management Plane Protection for Inband Interface, on page 184
- Configure Device for Management Plane Protection for Out-of-band Interface, on page 187
- Information About Implementing Management Plane Protection, on page 191

Benefits of Management Plane Protection

Implementing the MPP feature provides the following benefits:

- Greater access control for managing a device than allowing management protocols on all interfaces.
- Improved performance for data packets on non-management interfaces.
- Support for network scalability.
- Simplifies the task of using per-interface access control lists (ACLs) to restrict management access to the device.
- Fewer ACLs are needed to restrict access to the device.
- Prevention of packet floods on switching and routing interfaces from reaching the CPU.

Restrictions for Implementing Management Plane Protection

The following restrictions are listed for implementing Management Plane Protection (MPP):

- Currently, MPP does not keep track of the denied or dropped protocol requests.
- MPP configuration does not enable the protocol services. MPP is responsible only for making the services available on different interfaces. The protocols are enabled explicitly.
- Management requests that are received on inband interfaces are not necessarily acknowledged there.
- The changes made for the MPP configuration do not affect the active sessions that are established before the changes.
- Currently, MPP controls only the incoming management requests for protocols, such as TFTP, Telnet, Simple Network Management Protocol (SNMP), Secure Shell (SSH), XML and Netconf.
- MPP does not support MIB.

Configure Device for Management Plane Protection for Inband Interface

An *inband management interface* is a physical or logical interface that processes management packets, as well as data-forwarding packets. An inband management interface is also called a *shared management interface*. Perform this task to configure a device that you have just added to your network or a device already operating in your network. This task shows how to configure MPP as an inband interface in which Telnet is allowed to access the router only through a specific interface.

Perform the following additional tasks to configure an inband MPP interface in non-default VRF.

- Configure the interface under the non-default inband VRF.
- Configure the global inband VRF.
- In the case of Telnet, configure the Telnet VRF server for the inband VRF.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 control-plane

Example:

```
RP/0/RP0/CPU0:router(config)# control-plane
RP/0/RP0/CPU0:router(config-ctrl)#
```

Enters control plane configuration mode.

Step 3 management-plane

Example:

```
RP/0/RP0/CPU0:router(config-ctrl)# management-plane
RP/0/RP0/CPU0:router(config-mpp)#
```

Configures management plane protection to allow and disallow protocols and enters management plane protection configuration mode.

Step 4 inband

Example:

```
RP/0/RP0/CPU0:router(config-mpp)# inband
RP/0/RP0/CPU0:router(config-mpp-inband)#
```

Configures an inband interface and enters management plane protection inband configuration mode.

Step 5 **interface** {*type instance* | **all**}

Example:

```
RP/0/RP0/CPU0:router(config-mpp-inband)# interface HundredGigE 0/9/0/0
RP/0/RP0/CPU0:router(config-mpp-inband-Gi0_0_1_0)#
```

Configures a specific inband interface, or all inband interfaces. Use the **interface** command to enter management plane protection inband interface configuration mode.

- Use the **all** keyword to configure all interfaces.

Step 6 **allow** {*protocol* | **all**} [**peer**]

Example:

```
RP/0/RP0/CPU0:router(config-mpp-inband-Gi0_0_1_0)# allow Telnet peer
RP/0/RP0/CPU0:router(config-telnet-peer)#
```

Configures an interface as an inband interface for a specified protocol or all protocols.

- Use the *protocol* argument to allow management protocols on the designated management interface.
  - SNMP (also versions)
  - Secure Shell (v1 and v2)

- TFTP
- Telnet
- Netconf
- XML

- Use the **all** keyword to configure the interface to allow all the management traffic that is specified in the list of protocols.
- (Optional) Use the **peer** keyword to configure the peer address on the interface.

Step 7 **address ipv4** {*peer-ip-address* | *peer ip-address/length*}

Example:

```
RP/0/RP0/CPU0:router(config-telnet-peer)# address ipv4 10.1.0.0/16
```

Configures the peer IPv4 address in which management traffic is allowed on the interface.

- Use the *peer-ip-address* argument to configure the peer IPv4 address in which management traffic is allowed on the interface.
- Use the *peer ip-address/length* argument to configure the prefix of the peer IPv4 address.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 9 **show mgmt-plane** [**inband** |] [**interface** {*type instance*}]

Example:

```
RP/0/RP0/CPU0:router# show mgmt-plane inband interface HundredGigE 0/9/0/0
```

Displays information about the management plane, such as type of interface and protocols enabled on the interface.

- (Optional) Use the **inband** keyword to display the inband management interface configurations that are the interfaces that process management packets as well as data-forwarding packets.
- (Optional) Use the **interface** keyword to display the details for a specific interface.

Configure Device for Management Plane Protection for Out-of-band Interface

Out-of-band refers to an interface that allows only management protocol traffic to be forwarded or processed. An *out-of-band management interface* is defined by the network operator to specifically receive network management traffic. The advantage is that forwarding (or customer) traffic cannot interfere with the management of the router, which significantly reduces the possibility of denial-of-service attacks.

Out-of-band interfaces forward traffic only between out-of-band interfaces or terminate management packets that are destined to the router. In addition, the out-of-band interfaces can participate in dynamic routing protocols. The service provider connects to the router's out-of-band interfaces and builds an independent overlay management network, with all the routing and policy tools that the router can provide.

Perform the following tasks to configure an out-of-band MPP interface.

- Configure the interface under the out-of-band VRF.
- Configure the global out-of-band VRF.
- In the case of Telnet, configure the Telnet VRF server for the out-of-band VRF.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 control-plane

Example:

```
RP/0/RP0/CPU0:router(config)# control-plane
RP/0/RP0/CPU0:router(config-ctrl)#
```

Enters control plane configuration mode.

Step 3 management-plane

Example:

```
RP/0/RP0/CPU0:router(config-ctrl)# management-plane
RP/0/RP0/CPU0:router(config-mpp)#
```

Configures management plane protection to allow and disallow protocols and enters management plane protection configuration mode.

Step 4 out-of-band

Example:

```
RP/0/RP0/CPU0:router(config-mpp)# out-of-band
RP/0/RP0/CPU0:router(config-mpp-outband)#
```

Configures out-of-band interfaces or protocols and enters management plane protection out-of-band configuration mode.

Step 5 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-mpp-outband)# vrf target
```

Configures a Virtual Private Network (VPN) routing and forwarding (VRF) reference of an out-of-band interface.

- Use the *vrf-name* argument to assign a name to a VRF.

Step 6 **interface** {*type instance* | **all**}

Example:

```
RP/0/RP0/CPU0:router(config-mpp-outband)# interface HundredGigE 0/9/0/0
RP/0/RP0/CPU0:router(config-mpp-outband-if)#
```

Configures a specific out-of-band interface, or all out-of-band interfaces, as an out-of-band interface. Use the **interface** command to enter management plane protection out-of-band configuration mode.

- Use the **all** keyword to configure all interfaces.

Step 7 **allow** {*protocol* | **all**} [**peer**]

Example:

```
RP/0/RP0/CPU0:router(config-mpp-outband-if)# allow TFTP peer
RP/0/RP0/CPU0:router(config-tftp-peer)#
```

Configures an interface as an out-of-band interface for a specified protocol or all protocols.

- Use the *protocol* argument to allow management protocols on the designated management interface.
  - HTTP or HTTPS
  - SNMP (also versions)
  - Secure Shell (v1 and v2)
  - TFTP
  - Telnet

- Netconf

- Use the **all** keyword to configure the interface to allow all the management traffic that is specified in the list of protocols.
- (Optional) Use the **peer** keyword to configure the peer address on the interface.

Step 8 **address ipv6** {*peer-ip-address* | *peer ip-address/length*}

Example:

```
RP/0/RP0/CPU0:router(config-tftp-peer)# address ipv6 33::33
```

Configures the peer IPv6 address in which management traffic is allowed on the interface.

- Use the *peer-ip-address* argument to configure the peer IPv6 address in which management traffic is allowed on the interface.
- Use the *peer ip-address/length* argument to configure the prefix of the peer IPv6 address.

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show mgmt-plane** [**inband** | **out-of-band**] [**interface** {*type instance*} | **vrf**]

Example:

```
RP/0/RP0/CPU0:router# show mgmt-plane out-of-band interface HundredGigE 0/9/0/0
```

Displays information about the management plane, such as type of interface and protocols enabled on the interface.

- (Optional) Use the **inband** keyword to display the inband management interface configurations that are the interfaces that process management packets as well as data-forwarding packets.
- (Optional) Use the **out-of-band** keyword to display the out-of-band interface configurations.
- (Optional) Use the **interface** keyword to display the details for a specific interface.
- (Optional) Use the **vrf** keyword to display the Virtual Private Network (VPN) routing and forwarding reference of an out-of-band interface.

Example

The following example shows how to configure inband and out-of-band interfaces for a specific IP address under MPP:

```
configure
 control-plane
  management-plane
   inband
    interface all
     allow SSH
     !
    interface HundredGigE 0/9/0/0
     allow all
     allow SSH
     allow Telnet peer
      address ipv4 10.1.0.0/16
    !
   !
    interface HundredGigE 0/9/0/0
     allow Telnet peer
      address ipv4 10.1.0.0/16
     !
    !
   !
   out-of-band
    vrf my_out_of_band
    interface HundredGigE 0/9/0/0
     allow TFTP peer
      address ipv6 33::33
     !
    !
   !
  !
 !

show mgmt-plane

Management Plane Protection

inband interfaces
----------------------

interface - HundredGigE0_9_0_0
        ssh configured -
                All peers allowed
        telnet configured -
                peer v4 allowed - 10.1.0.0/16
        all configured -
                All peers allowed
interface - HundredGigE0_9_0_0
        telnet configured -
                peer v4 allowed - 10.1.0.0/16

interface - all
        all configured -
                All peers allowed

outband interfaces
----------------------
interface - HundredGigE0_9_0_0
```

```
        tftp configured -
                peer v6 allowed - 33::33

show mgmt-plane out-of-band vrf

Management Plane Protection -
        out-of-band VRF - my_out_of_band
```

Information About Implementing Management Plane Protection

Before you enable the Management Plane Protection feature, you should understand the following concepts:

Peer-Filtering on Interfaces

The peer-filtering option allows management traffic from specific peers, or a range of peers, to be configured.

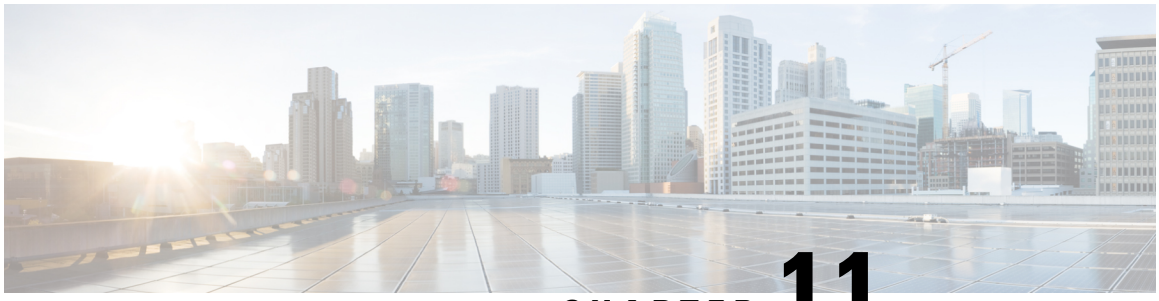
Control Plane Protection

A *control plane* is a collection of processes that run at the process level on a route processor and collectively provide high-level control for most Cisco software functions. All traffic directly or indirectly destined to a router is handled by the control plane. Management Plane Protection operates within the Control Plane Infrastructure.

Management Plane

The *management plane* is the logical path of all traffic that is related to the management of a routing platform. One of three planes in a communication architecture that is structured in layers and planes, the management plane performs management functions for a network and coordinates functions among all the planes (management, control, and data). In addition, the management plane is used to manage a device through its connection to the network.

Examples of protocols processed in the management plane are Simple Network Management Protocol (SNMP), Telnet, SSH, XML and Netconf. These management protocols are used for monitoring and for command-line interface (CLI) access. Restricting access to devices to internal sources (trusted networks) is critical.



CHAPTER 11

Traffic Protection for Third-Party Applications

Traffic Protection for Third-Party Applications provides a mechanism for securing management traffic on the router. Without Traffic Protection for Third-Party Applications, if the service is enabled, the Cisco IOS XR allows the service traffic to pass through any interface with a network address.

Traffic Protection for Third-Party Applications helps in rate limiting or throttling the traffic through configuration with the help of LPTS. Traffic Protection for Third-Party Applications filters traffic based on the following tuples: address family, vrf, port, interface, local address and remote address.

Note It is mandatory to configure address family, protocol, local port, and vrf, as well as at least one of interface or local or remote address.

- Example: Traffic Protection for Third-Party Applications over gRPC, on page 193
- Limitations for Traffic Protection for Third-Party Applications over gRPC, on page 194
- Prerequisites for Traffic Protection for Third-Party Applications vver gRPC , on page 194
- Configuring Traffic Protection for Third-Party Applications, on page 194
- Troubleshooting Traffic Protection for Third-Party Applications, on page 195

Example: Traffic Protection for Third-Party Applications over gRPC

This section explains how to use Traffic Protection for Third-Party Applications (TPA) over gRPC through the gRPC port 57600. Once gRPC is enabled, the router has a TPA and you can apply the traffic protection rules for the gRPC TPA.

Google-defined Remote Procedure Calls (gRPC) is an open-source RPC framework. It is based on Protocol Buffers (Protobuf), which is an open source binary serialization protocol. gRPC provides a flexible, efficient, automated mechanism for serializing structured data, like XML, but is smaller and simpler to use. The user needs to define the structure by defining protocol buffer message types in .proto files. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs.

Cisco gRPC Interface Definition Language (IDL) uses a set of supported RPCs such as get-config, merge-config, replace-config, cli-config, delete-config, cli-show, get-models, action-json, commit, and commit-replace. gRPC server runs in Extensible Manageability Services Daemon (emsd) process. gRPC client can be on any machine.

gRPC encodes requests and responses in binary. gRPC is extensible to other content types along with Protobuf. The Protobuf binary data object in gRPC is transported over HTTP/2.

Note It is recommended to configure TLS before enabling gRPC. Enabling gRPC protocol uses the default HTTP/2 transport with no TLS enabled on TCP. gRPC mandates AAA authentication and authorization for all gRPC requests. If TLS is not configured, the authentication credentials are transferred over the network unencrypted. Non-TLS mode can only be used in secure internal network.

gRPC supports distributed applications and services between a client and server. gRPC provides the infrastructure to build a device management service to exchange configuration and operational data between a client and a server. The structure of the data is defined by YANG models. For more information, see *Use gRPC Protocol to Define Network Operations with Data Models* in the *Programmability Configuration Guide*.

Limitations for Traffic Protection for Third-Party Applications over gRPC

The following limitations are applicable for the Traffic Protection for Third-Party Applications:

- If the TPA entry is configured with only the active RP management interface and redundancy switchover is performed, the gRPC connection fails.

Prerequisites for Traffic Protection for Third-Party Applications vver gRPC

Ensure that the gRPC is configured.

gRPC Configuration

```
Router(config)# grpc port port-number
Router(config)# grpc no-tls
Router(config-grpc)# commit
```

Running Configuration

```
Router# show running-config grpc

grpc port 57600
 no-tls
!
```

Configuring Traffic Protection for Third-Party Applications

The following task shows how to configure traffic protection for third-party applications

```
RP/0/0/CPU0:ios#configure
RP/0/0/CPU0:ios(config)#tpa
```

```
RP/0/0/CPU0:ios(config-tpa)#vrf default
RP/0/0/CPU0:ios(config-tpa-vrf)#address-family ipv4
RP/0/0/CPU0:ios(config-tpa-vrf-afi)#protection
RP/0/0/CPU0:ios(config-tpa-vrf-afi-prot)#allow protocol {udp|tcp} local-port {port-number}
 remote-address {remote-ip-address} local-address {local-ip-address}
```

Running Configuration

```
Router# show running-config
tpa
vrf default
address-family ipv4
protection
allow protocol tcp local-port 57600 remote-address 10.0.0.2/32 local-address 192.168.0.1/32
allow protocol tcp local-port 57600 remote-address 10.0.1.0/24 local-address 192.168.0.1/32
allow protocol tcp local-port 57600 remote-address 10.0.2.0/24 local-address 192.168.0.1/32
address-family ipv6
protection
allow protocol tcp local-port 57600 remote-address 2001:DB8::1/128 local-address
2001:DB8:0:ABCD::1/128
allow protocol tcp local-port 57600 remote-address 2001:DB8::2/128 local-address
2001:DB8:0:ABCD::1/128
allow protocol tcp local-port 57600 remote-address 2001:DB8::3/128 local-address
2001:DB8:0:ABCD::1/128
!
   !
  !
!
```

For more information on **tpa** and **allow local-port** commands, see *Management Plane Protection Commands* Chapter of the *System Security Command Reference for Cisco NCS 5500 Series, Cisco NCS 540 Series, and Cisco NCS 560 Series Routers*.

Troubleshooting Traffic Protection for Third-Party Applications

The following show command output displays the TPA configuration.

```
Router# show running-config tpa


  tpa
   vrf default
    address-family ipv4
    allow local-port 57600 protocol tcp inter mgmtEth 0/RP0/CPU0/0 local-address
192.168.0.1/32 remote-address 10.0.0.2/32
    !
```

gRPC Configuration without TPA

```
Router# show kim lpts database

State:
 Prog - Programmed in hardware
 Cfg  - Configured, not yet programmed
 Ovr  - Not programmed, overridden by user configuration
 Intf - Not programmed, interface does not exist

Owner  AF Proto State       Interface      VRF          Local ip,port > Remote ip,port
------ -- ----- ----- -------------- ------------ ------------------------------
Linux   2     6          Prog                           global-vrf            any,57600
 > any,0
```

```
Router# show lpts bindings brief | include TPA
 0/RP0/CPU0 TPA  LR IPV4 TCP    default   any          any,57600 any
```

gRPC Configuration with TPA

The following show command output displays the things that are configured in the LPTS database. It also checks if gRPC configuration is owned by Linux without using any filters.

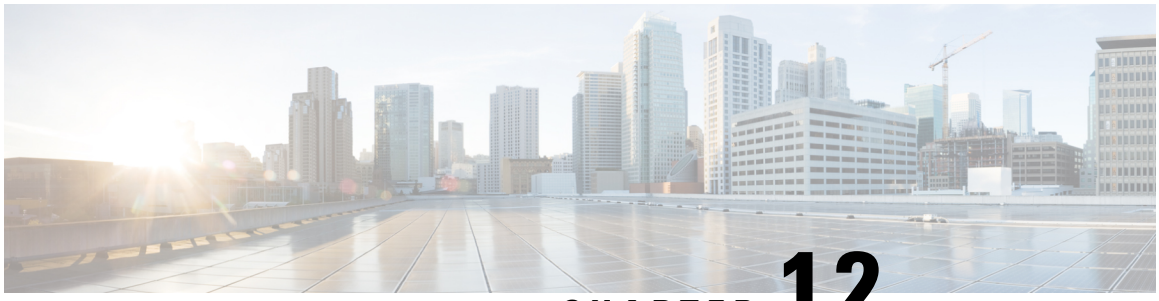
```
Router# show kim lpts database

State:
 Prog - Programmed in hardware
 Cfg  - Configured, not yet programmed
 Ovr  - Not programmed, overridden by user configuration
 Intf - Not programmed, interface does not exist

Owner  AF Proto State Interface      VRF          Local ip,port > Remote ip,port
------ -- ----- ----- -------------- ------------ ------------------------------
Client  2     6 Prog                 default      192.168.0.1/32,57600  > 10.0.0.2/32,0
Linux   2     6 Ovr                  global-vrf   any,57600  > any,0
```

```
Router# show lpts bindings brief | include TPA

 0/RP0/CPU0 TPA  LR IPV4 TCP    default   Mg0/RP0/CPU0/0 192.168.0.1,57600 10.0.0.2
Router#
Router# 0/RP0/ADMIN0:Mar 19 15:22:26.837 IST: pm[2433]:
%INFRA-Process_Manager-3-PROCESS_RESTART : Process tams (IID: 0) restarted
```



CHAPTER 12

Implementing Secure Shell

Secure Shell (SSH) is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley **rexec** and **rsh** tools.

Two versions of the SSH server are available: SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2). SSHv1 uses Rivest, Shamir, and Adelman (RSA) keys and SSHv2 uses either Digital Signature Algorithm (DSA) keys or Rivest, Shamir, and Adelman (RSA) keys, or Elliptic Curve Digital Signature Algorithm (ECDSA) keys. Cisco software supports both SSHv1 and SSHv2.

This module describes how to implement Secure Shell.

- Implementing Secure Shell, on page 197
- SSH Port Forwarding, on page 235
- Non-Default SSH Port, on page 238
- Multi-Factor Authentication for SSH, on page 242
- SSH server timeouts, on page 247

Implementing Secure Shell

Secure Shell (SSH) is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley **rexec** and **rsh** tools.

Two versions of the SSH server are available: SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2). SSHv1 uses Rivest, Shamir, and Adelman (RSA) keys and SSHv2 uses either Digital Signature Algorithm (DSA) keys or Rivest, Shamir, and Adelman (RSA) keys, or Elliptic Curve Digital Signature Algorithm (ECDSA) keys. Cisco software supports both SSHv1 and SSHv2.

This module describes how to implement Secure Shell.

Information About Implementing Secure Shell

To implement SSH, you should understand the following concepts:

SSH Server

The SSH server feature enables an SSH client to make a secure, encrypted connection to a Cisco router. This connection provides functionality that is similar to that of an inbound Telnet connection. Before SSH, security

was limited to Telnet security. SSH allows a strong encryption to be used with the Cisco software authentication. The SSH server in Cisco software works with publicly and commercially available SSH clients.

SSH Client

The SSH client feature is an application running over the SSH protocol to provide device authentication and encryption. The SSH client enables a Cisco router to make a secure, encrypted connection to another Cisco router or to any other device running the SSH server. This connection provides functionality that is similar to that of an outbound Telnet connection except that the connection is encrypted. With authentication and encryption, the SSH client allows for a secure communication over an insecure network.

The SSH client works with publicly and commercially available SSH servers. The SSH client supports the ciphers of AES, 3DES, message digest algorithm 5 (MD5), SHA1, and password authentication. User authentication is performed in the Telnet session to the router. The user authentication mechanisms supported for SSH are RADIUS, TACACS+, and the use of locally stored usernames and passwords.

The SSH client supports setting DSCP value in the outgoing packets.

```
ssh client dscp <value from 0 - 63>
```

If not configured, the default DSCP value set in packets is 16 (for both client and server).

The SSH client supports the following options:

- DSCP—DSCP value for SSH client sessions.

```
RP/0/5/CPU0:router#configure
RP/0/5/CPU0:router(config)#ssh ?
  client   Provide SSH client service
  server   Provide SSH server service
  timeout  Set timeout value for SSH
RP/0/5/CPU0:router(config)#ssh client ?
```

- Knownhost—Enable the host pubkey check by local database.
- Source-interface—Source interface for SSH client sessions.

```
RP/0/5/CPU0:router(config)#ssh client source-interface ?
  ATM                 ATM Network Interface(s)
  BVI                 Bridge-Group Virtual Interface
  Bundle-Ether        Aggregated Ethernet interface(s)
  CEM                 Circuit Emulation interface(s)
  GigabitEthernet     GigabitEthernet/IEEE 802.3 interface(s)
  IMA                 ATM Network Interface(s)
  IMtestmain          IM Test Interface
  Loopback            Loopback interface(s)
  MgmtEth             Ethernet/IEEE 802.3 interface(s)
  Multilink           Multilink network interface(s)
  Null                Null interface
  PFItestmain         PFI Test Interface
  PFItestnothw        PFI Test Not-HW Interface
  PW-Ether            PWHE Ethernet Interface
  PW-IW               PWHE VC11 IP Interworking Interface
  Serial              Serial network interface(s)
  VASILeft            VASI Left interface(s)
  VASIRight           VASI Right interface(s)
  test-bundle-channel  Aggregated Test Bundle interface(s)
  tunnel-ipsec        IPSec Tunnel interface(s)
  tunnel-mte          MPLS Traffic Engineering P2MP Tunnel interface(s)
  tunnel-te           MPLS Traffic Engineering Tunnel interface(s)
  tunnel-tp           MPLS Transport Protocol Tunnel interface
RP/0/5/CPU0:router(config)#ssh client source-interface
RP/0/5/CPU0:router(config)#
```

SSH also supports remote command execution as follows:

```
RP/0/5/CPU0:router#ssh ?
  A.B.C.D  IPv4 (A.B.C.D) address
  WORD     Hostname of the remote node
  X:X::X   IPv6 (A:B:C:D...:D) address
  vrf      vrf table for the route lookup
RP/0/5/CPU0:router#ssh 10.1.1.1 ?
  cipher            Accept cipher type
  command           Specify remote command (non-interactive)
  source-interface  Specify source interface
  username          Accept userid for authentication
  <cr>
RP/0/5/CPU0:router#ssh 192.68.46.6 username admin command "show redundancy sum"
Password:


Wed Jan  9 07:05:27.997 PST
    Active Node    Standby Node
    -----------    ------------
       0/4/CPU0        0/5/CPU0 (Node Ready, NSR: Not Configured)

RP/0/5/CPU0:router#
```

SFTP Feature Overview

SSH includes support for standard file transfer protocol (SFTP) , a new standard file transfer protocol introduced in SSHv2. This feature provides a secure and authenticated method for copying router configuration or router image files.

The SFTP client functionality is provided as part of the SSH component and is always enabled on the router. Therefore, a user with the appropriate level can copy files to and from the router. Like the **copy** command, the **sftp** command can be used only in XR EXEC mode.

The SFTP client is VRF-aware, and you may configure the secure FTP client to use the VRF associated with a particular source interface during connections attempts. The SFTP client also supports interactive mode, where the user can log on to the server to perform specific tasks via the Unix server.

The SFTP Server is a sub-system of the SSH server. In other words, when an SSH server receives an SFTP server request, the SFTP API creates the SFTP server as a child process to the SSH server. A new SFTP server instance is created with each new request.

The SFTP requests for a new SFTP server in the following steps:

- The user runs the **sftp** command with the required arguments
- The SFTP API internally creates a child session that interacts with the SSH server
- The SSH server creates the SFTP server child process
- The SFTP server and client interact with each other in an encrypted format
- The SFTP transfer is subject to LPTS policer "SSH-Known". Low policer values will affect SFTP transfer speeds

Note In IOS-XR SW release 4.3.1 onwards the default policer value for SSH-Known has been reset from 2500pps to 300pps. Slower transfers are expected due to this change. You can adjust the lpts policer value for this punt cause to higher values that will allow faster transfers

When the SSH server establishes a new connection with the SSH client, the server daemon creates a new SSH server child process. The child server process builds a secure communications channel between the SSH client and server via key exchange and user authentication processes. If the SSH server receives a request for the sub-system to be an SFTP server, the SSH server daemon creates the SFTP server child process. For each incoming SFTP server subsystem request, a new SSH server child and a SFTP server instance is created. The SFTP server authenticates the user session and initiates a connection. It sets the environment for the client and the default directory for the user.

Once the initialization occurs, the SFTP server waits for the SSH_FXP_INIT message from the client, which is essential to start the file communication session. This message may then be followed by any message based on the client request. Here, the protocol adopts a 'request-response' model, where the client sends a request to the server; the server processes this request and sends a response.

The SFTP server displays the following responses:

- Status Response
- Handle Response
- Data Response
- Name Response

Note The server must be running in order to accept incoming SFTP connections.

RSA Based Host Authentication

Verifying the authenticity of a server is the first step to a secure SSH connection. This process is called the host authentication, and is conducted to ensure that a client connects to a valid server.

The host authentication is performed using the public key of a server. The server, during the key-exchange phase, provides its public key to the client. The client checks its database for known hosts of this server and the corresponding public-key. If the client fails to find the server's IP address, it displays a warning message to the user, offering an option to either save the public key or discard it. If the server’s IP address is found, but the public-key does not match, the client closes the connection. If the public key is valid, the server is verified and a secure SSH connection is established.

The IOS XR SSH server and client had support for DSA based host authentication. But for compatibility with other products, like IOS, RSA based host authentication support is also added.

RSA Based User Authentication

One of the method for authenticating the user in SSH protocol is RSA public-key based user authentication. The possession of a private key serves as the authentication of the user. This method works by sending a signature created with a private key of the user. Each user has a RSA keypair on the client machine. The private key of the RSA keypair remains on the client machine.

The user generates an RSA public-private key pair on a unix client using a standard key generation mechanism such as ssh-keygen. The max length of the keys supported is 4096 bits, and the minimum length is 512 bits. The following example displays a typical key generation activity:

```
bash-2.05b$ ssh-keygen -b 1024 -t rsa
Generating RSA private key, 1024 bit long modulus
```

The public key must be in base64 encoded (binary) formats for it to be imported correctly into the router.

Note You can use third party tools available on the Internet to convert the key to the binary format.

Once the public key is imported to the router, the SSH client can choose to use the public key authentication method by specifying the request using the "-o" option in the SSH client. For example:

```
client$ ssh -o PreferredAuthentications=publickey 1.2.3.4
```

If a public key is not imported to a router using the RSA method, the SSH server initiates the password based authentication. If a public key is imported, the server proposes the use of both the methods. The SSH client then chooses to use either method to establish the connection. The system allows only 10 outgoing SSH client connections.

Currently, only SSH version 2 and SFTP server support the RSA based authentication.

Note The preferred method of authentication would be as stated in the SSH RFC. The RSA based authentication support is only for local authentication, and not for TACACS/RADIUS servers.

Authentication, Authorization, and Accounting (AAA) is a suite of network security services that provide the primary framework through which access control can be set up on your Cisco router or access server.

SSHv2 Client Keyboard-Interactive Authentication

An authentication method in which the authentication information is entered using a keyboard is known as keyboard-interactive authentication. This method is an interactive authentication method in the SSH protocol. This type of authentication allows the SSH client to support different methods of authentication without having to be aware of their underlying mechanisms.

Currently, the SSHv2 client supports the keyboard-interactive authentication. This type of authentication works only for interactive applications.

Note The password authentication is the default authentication method. The keyboard-interactive authentication method is selected if the server is configured to support only the keyboard-interactive authentication.

Prerequisites for Implementing Secure Shell

The following prerequisites are required to implement Secure Shell:

- Download the required image on your router. The SSH server and SSH client require you to have a a crypto package (data encryption standard [DES], 3DES and AES) from Cisco downloaded on your router.

Note From Cisco IOS XR Software Release 7.0.1 and later, the SSH and SFTP components are available in the baseline Cisco IOS XR software image itself. For details, see, SSH and SFTP in Baseline Cisco IOS XR Software Image, on page 202.

- Configure user authentication for local or remote access. You can configure authentication with or without authentication, authorization, and accounting (AAA).
- AAA authentication and authorization must be configured correctly for Secure Shell File Transfer Protocol (SFTP) to work.

SSH and SFTP in Baseline Cisco IOS XR Software Image

From Cisco IOS XR Software Release 7.0.1 and later, the management plane and control plane components that were part of the Cisco IOS XR security package (k9sec package) are moved to the base Cisco IOS XR software image. These include SSH, SCP and SFTP. However, 802.1X protocol (Port-Based Network Access Control) and data plane components remain as a part of the security package as per the export compliance regulations.This segregation of package components makes the software more modular. It also gives you the flexibility of including or excluding the security package as per your requirements.

The base package and the security package allow FIPS, so that the control plane can negotiate FIPS-approved algorithms.

FIDO2 authentication for SSH

A FIDO2 authentication for SSH is an open authentication standard that

- enables secure, passwordless SSH logins and multi-factor authentication on Cisco IOS XR devices,
- uses hardware security keys to store private keys, and
- requires user presence such as touching the key to complete the login process.

Unlike typical SSH key pair generation, where both keys are stored on the ssh client, with FIDO2 the private key remains only on the hardware device. The public key algorithms commonly used are `ed25519-sk` and `ecdsa-sk`.

In SSH, public key authentication works by storing the public key on the host device. The SSH client sends the public key during authentication, and the server verifies if the key exists in the host database for that user. Passwordless authentication is only permitted if the keys are registered.

Table 22: Feature History Table

Feature Name	Release Information	Feature Description
FIDO2 authentication for SSH	Release 26.2.1	FIDO2 support for SSH enables secure, passwordless logins by using hardware security keys to store private keys. This feature requires physical user presence to authenticate sessions and protects against phishing by verifying a signature from the security device during the login process.

Restrictions for configuring FIDO2 authentication for SSH

- Remote authentication using TACACS+ or RADIUS is not supported for FIDO2 keys.
- FIDO2 public keys cannot be stored in cepki sysdb.
- XRSSH doesn't support FIDO2 authentication.
- Only ed25519-sk and ecdsa-sk key types are supported
- Automation of full authentication flow is limited due to mandatory user presence.
- FIDO2 keys cannot be imported using `crypto key import authentication`.

Guidelines for configuring FIDO support for SSH

- FIDO2 authentication is supported only for local SSH public key authentication.
- Public keys must be stored in the authorized_keys file.
- Use the existing ssh-server authorized-keys CLI to import FIDO2 public keys.
- The private key never leaves the FIDO2 device.
- User presence (touch/biometric) is mandatory during SSH authentication.
- Multiple FIDO2 public keys can be imported for a single user.

Configure FIDO2 authentication for SSH

Set up FIDO2-based SSH authentication on Cisco IOS XR routers, enabling secure logins using a hardware security key such as a YubiKey.

FIDO2 authentication strengthens device access security by requiring both the user’s credentials and physical presence of a hardware-backed authenticator. This protects against phishing attacks and credential theft. Use this method when you need to enhance SSH security for administrators or automation systems accessing Cisco IOS XR routers, or when organizational policy mandates strong, multi-factor authentication.

Before you begin

- Ensure you have a supported client - macOS, Windows, or any OS running OpenSSH 8.2+.
- Obtain a FIDO2-compatible device such as a YubiKey.
- Access privileges on the target Cisco IOS XR router.

Procedure

Step 1 Connect your FIDO2 device to the client machine.

Plug in your YubiKey to your client (macOS, Windows, or supported system).

Step 2 Generate FIDO2 key pairs on your client.

Example:

```
ssh-keygen -t ecdsa-sk -f id_ecdsa_sk
```

a) In your terminal, run **ssh-keygen -t ecdsa-sk -f id_ecdsa_sk** command.
b) When prompted, touch your authenticator of YubiKey to authorize key generation.
c) Enter a passphrase for the key if desired, then confirm the passphrase.
d) A public key (`id_ecdsa_sk.pub`) and private key will be generated.

Step 3 Import your public key to the IOS XR router.

a) Copy the contents of `id_ecdsa_sk.pub` from your client.
b) On the router, enter EXEC mode and run the **ssh-server authorized-keys** command.
c) If you do not provide a username, the key is imported for the current logged-in user.
d) Paste the public key when prompted.

Step 4 Verify the imported SSH key on the IOS XR router.

Example:

```
Router#show ssh server authorized-keys user <username>
```

a) Confirm your key appears in the authorized keys list.

Step 5 Test SSH access using the FIDO2 key.

Example:

```
Router#show ssh
SSH version : Cisco-2.0
id       chan pty     location        state           userid    host                  ver
authentication connection type
-------------------------------------------------------------------------------------------
Incoming sessions
974       0   vty1    0/RP0/CPU0      SESSION_OPEN    cafyauto  10.189.219.24         v2
sk-ecdsa-sha2-nistp256@openssh Command-Line-Interface
```

a) On your client, use the command:

Example:

```
ssh -i id_ecdsa_sk <username>@<router_ip>
```

b) When prompted, touch your YubiKey to confirm presence.
c) Upon successful authentication, you will see a login confirmation.

Step 6 (Optional) Import multiple public keys for a user.

You can import additional keys using step 3 for a specific username; the behavior remains as with previous (non-FIDO) public keys.

You have successfully configured FIDO-based SSH authentication for enhanced security. SSH connections now require your FIDO device for login.

Netconf access controls

A Netconf access control is a security mechanism that

- blocks Netconf requests on the SSH port (default 22) and allows only on the designated Netconf port (default 830)
- restricts Netconf access based on specified IP addresses using Access Control Lists (ACLs), and
- allows other SSH services such as Secure Copy Protocol (SCP) and Secure File Transfer Protocol (SFTP) to continue functioning.

Table 23: Feature history table

Cisco IOS XR router uses Netconf over SSH. By default, SSH uses port 22 and Netconf uses port 830. Netconf requests are permitted on the SSH port by default.

Without this feature enabled, a NETCONF connection can be established over the SSH port even from addresses that are not permitted by the ACLs.

Benefits of Netconf access control

This feature provides these benefits:

- You can block Netconf for specific IP addresses while still allowing SSH access for those addresses.
- You can prevent unauthorized Netconf access on the SSH port.
- You retain access to other SSH services such as SCP and SFTP.
- You enforce Netconf access restrictions without disrupting SSH access.

Best practice for Netconf access control

- Use ACLs to specify IP addresses that you block for Netconf.
- Apply the **ssh server netconf disable ssh-port** to disable Netconf on the SSH port.

Restrictions for Netconf access control

- During the initial TCP handshake and authentication, you cannot identify the service from the SSH protocol. At the channel request stage, the server checks for a Netconf request and then rejects your connection request.
- Because the SSH server does not identify the request type until the channel request phase, it cannot reject a Netconf connection on the SSH port by IP address during the initial connection. If an ACL blocks an IP address on the Netconf port, the connection attempt fails during the TCP handshake, and you cannot connect.
- Netconf access control blocks Netconf only on the SSH port for your device, but it does not block SCP or SFTP for the same IP address.

How Netconf access control works

Summary

Use Netconf access control to configure access to Netconf services through the SSH server and secure your device.

The components involved in the process are:

- **SSH server**: Manages incoming SSH connections and enforces access control.
- **SSH port**: Network port used for SSH and Netconf services.
- **Netconf service**: Provides configuration and management operations using the Netconf protocol.
- **Access Control Lists (ACLs)**: These restrict access to Netconf based on IP addresses.
- **Users and clients**: They attempt to access device configuration using Netconf.

Workflow

The SSH server enforces Netconf restrictions on the SSH port using this process:

1. The administrator configures the SSH server to disable Netconf access on the SSH port using the **ssh server netconf disable ssh-port** command.
2. When the server receives a NETCONF request, it verifies that the request is received on a port configured for SSH.
3. After the SSH session is established, the server checks the requested service type.
4. If NETCONF is requested on an SSH port where NETCONF has been disabled through the CLI, the server rejects the request and terminates the session.
5. SSH continues to allow other services for the same IP address, including SCP and SFTP.
6. This behavior runs in XR SSH and CiscoSSH components.

Result

Your configuration restricts Netconf access and allows permitted SSH functions.

Configure Netconf access control

Restrict Netconf access over SSH to enhance device security and prevent unauthorized Netconf sessions.

Netconf is a network management protocol that operates over SSH. You may need to disable Netconf on the SSH port to harden device security or meet specific compliance requirements.

Before you begin

- Ensure you have administrative access to the device.
- Back up the current configuration before you make changes.
- Confirm that disabling Netconf over SSH will not interrupt management tasks if you use Netconf.

Follow these steps to configure Netconf access control:

Procedure

Step 1 Enter configuration mode.

Example:

```
Router#config
```

Step 2 Disable Netconf on the SSH port:

```
Router#ssh server netconf disable ssh-port
```

Note
Use the **ssh server netconf disable ssh-port** configuration for any configured SSH port (default port 22 or any custom port).

Step 3 Commit the configuration.

Example:

```
Router#commit
```

Syslogs for Netconf access control

You can observe these SYSLOGS on the router console:

```
RP/0/RP0/CPU0:Nov 20 07:55:09.476 UTC: ssh_syslog_proxy[1221]:
%SECURITY-SSHD_SYSLOG_PRX-3-ERR_GENERAL : sshd[33048]: On ssh port invalid netconf channel
 request received

RP/0/RP0/CPU0:Nov 20 07:55:09.476 UTC: ssh_syslog_proxy[1221]:
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[33048]: subsystem request for netconf by
user cafyauto failed, subsystem not found
```

You have disabled Netconf access over SSH port. The device does not accept Netconf management sessions on the SSH port.

Guidelines and Restrictions for Implementing Secure Shell

The following are some basic SSH guidelines, restrictions, and limitations of the SFTP feature:

- In order for an outside client to connect to the router, the router needs to have an RSA (for SSHv1 or SSHv2) or DSA (for SSHv2) or ECDSA (for SSHv2) key pair configured. ECDSA, DSA and RSA keys are not required if you are initiating an SSH client connection from the router to an outside routing device. The same is true for SFTP: ECDSA, DSA and RSA keys are not required because SFTP operates only in client mode.
- If you delete all the default crypto keys (the keys with **the_default** label) on the router, the SSH clients cannot establish sessions with the router. Hence, for clients to successfully establish SSH sessions with the router, ensure that at least one default crypto key is always present on the router. In FIPS mode, it is mandatory to have at least one default crypto key of type RSA or ECDSA.
- For SSH sessions, the router supports key-exchange algorithms (**diffie-hellman-group1-sha1** and **curve25519**) and cipher algorithms (**3des-cbc** and **chacha20-poly1305@openssh.com**) only in non-FIPS mode. For routers supporting open source-based CiscoSSH, the SSH session fails to connect if any of

these algorithms is pre-configured prior to enabling FIPS mode. Whereas for routers supporting Cisco IOS XR SSH, the SSH session continues to connect in such scenarios.

Starting Cisco IOS XR Software Release 24.2.2, 24.3.2, and 24.4.1 and later, the SSH session fails to connect in such scenarios where these algorithms are pre-configured even for routers supporting Cisco IOS XR SSH.

- In order for SFTP to work properly, the remote SSH server must enable the SFTP server functionality. For example, the SSHv2 server is configured to handle the SFTP subsystem with a line such as **/etc/ssh2/sshd2_config**:
- **subsystem-sftp /usr/local/sbin/sftp-server**
- The SFTP server is usually included as part of SSH packages from public domain and is turned on by default configuration.
- SFTP is compatible with sftp server version OpenSSH_2.9.9p2 or higher.
- RSA-based user authentication is supported in the SSH and SFTP servers. The support however, is not extended to the SSH client.
- Execution shell and SFTP are the only applications supported.
- The SFTP client does not support remote filenames containing wildcards (* ?, []). The user must issue the **sftp** command multiple times or list all of the source files from the remote host to download them on to the router. For uploading, the router SFTP client can support multiple files specified using a wildcard provided that the issues mentioned in the first through third bullets in this section are resolved.
- The cipher preference for the SSH server follows the order AES128, AES192, AES256, and, finally, 3DES. The server rejects any requests by the client for an unsupported cipher, and the SSH session does not proceed.
- Use of a terminal type other than vt100 is not supported, and the software generates a warning message in this case.
- Password messages of “none” are unsupported on the SSH client.
- Files created on the local device lose the original permission information because the router infrastructure does not provide support for UNIX-like file permissions. For files created on the remote file system, the file permission adheres to the umask on the destination host and the modification and last access times are the time of the copy.

Configure SSH

Perform this task to configure SSH.



Note For SSHv1 configuration, Step 1 to Step 4 are required. For SSHv2 configuration, Step to Step 4 are optional.

Note From Cisco IOS XR Software Release 7.0.1 and later, the SSH host-key pairs are auto-generated at the time of router boot up. Hence you need not perform steps 5 to 7 to generate the host keys explicilty. See, #unique_244 for details.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **hostname** *hostname*

Example:

```
RP/0/RP0/CPU0:router(config)# hostname router1
```

Configures a hostname for your router.

Step 3 **domain name** *domain-name*

Example:

```
RP/0/RP0/CPU0:router(config)# domain name cisco.com
```

Defines a default domain name that the software uses to complete unqualified host names.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5 **crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*]

Example:

```
RP/0/RP0/CPU0:router# crypto key generate rsa general-keys
```

Generates an RSA key pair. The RSA key modulus can be in the range of 512 to 4096 bits.

- To delete the RSA key pair, use the **crypto key zeroize rsa** command.
- This command is used for SSHv1 only.

Step 6 crypto key generate dsa

Example:

```
RP/0/RP0/CPU0:router# crypto key generate dsa
```

Enables the SSH server for local and remote authentication on the router. The supported key sizes are: 512, 768 and 1024 bits.

- The recommended minimum modulus size is 1024 bits.
- Generates a DSA key pair.

  To delete the DSA key pair, use the **crypto key zeroize dsa** command.
- This command is used only for SSHv2.

Step 7 **crypto key generate ecdsa** [**nistp256** | **nistp384** | **nistp521**]

Example:

```
RP/0/RP0/CPU0:router# crypto key generate ecdsa nistp256
```

Generates an ECDSA key pair. The supported ECDSA curve types are: Nistp256, Nistp384 and Nistp521.

- To delete the ECDSA key pair, use the **crypto key zeroize ecdsa [nistp256 | nistp384 | nistp521]** command.
- This command is used for SSHv2 only.

Step 8 configure

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 9 **ssh timeout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config)# ssh timeout 60
```

(Optional) Configures the timeout value for user authentication to AAA.

- If the user fails to authenticate itself to AAA within the configured time, the connection is terminated.
- If no value is configured, the default value of 30 seconds is used. The range is from 5 to 120.

Step 10 Do one of the following:

- **ssh server** [**vrf** *vrf-name*]
- ssh server v2

Example:

```
RP/0/RP0/CPU0:router(config)# ssh server v2
```

- (Optional) Brings up an SSH server using a specified VRF of up to 32 characters. If no VRF is specified, the default VRF is used.

  To stop the SSH server from receiving any further connections for the specified VRF, use the **no** form of this command. If no VRF is specified, the default is assumed.

Note
The SSH server can be configured for multiple VRF usage.

- (Optional) Forces the SSH server to accept only SSHv2 clients if you configure the SSHv2 option by using the **ssh server v2** command. If you choose the **ssh server v2** command, only the SSH v2 client connections are accepted.

Step 11 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 12 show ssh

Example:

```
RP/0/RP0/CPU0:router# show ssh
```

(Optional) Displays all of the incoming and outgoing SSHv1 and SSHv2 connections to the router.

Step 13 show ssh session details

Example:

```
RP/0/RP0/CPU0:router# show ssh session details
```

(Optional) Displays a detailed report of the SSHv2 connections to and from the router.

Step 14 **show ssh history**

Example:

```
RP/0/RP0/CPU0:router# show ssh history
```

(Optional) Displays the last hundred SSH connections that were terminated.

Step 15 **show ssh history details**

Example:

```
RP/0/RP0/CPU0:router# show ssh history details
```

(Optional) Displays the last hundred SSH connections that were terminated with additional details. This command is similar to **show ssh session details** command but also mentions the start and end time of the session.

Step 16 **show tech-support ssh**

Example:

```
RP/0/RP0/CPU0:router# show tech-support ssh
```

(Optional) Automatically runs the `show` commands that display system information.

Note The order of priority while doing negotiation for a SSH connection is as follows:

1. ecdsa-nistp-521
2. ecdsa-nistp-384
3. ecdsa-nistp-256
4. rsa
5. dsa

SSH client strict host key checks

A SSH client strict host key checking is a feature that

- enforces validation of server host keys using a system-wide `known_hosts` file,
- restricts SSH client connections to hosts with trusted or previously accepted host keys, and
- supports configurable policies for new or changed keys, such as off, accept-new, ask, or yes.

The knownhosts file is located at `/mnt/rdsfs/ciscossh/known_hosts` and is shared across all users, providing persistent, centralized host key management for SSH client operations.

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
SSH client strict host key checking	Release 26.3.1	You enhance SSH security by enforcing strict host key checking, allowing you to control how the SSH client handles new or changed server keys - accept, reject, or prompt for approval. Trusted host keys are stored system-wide and persist across reloads, ensuring consistent validation for your outbound SSH connections.

Restrictions for SSH client strict host key check

Provides reference information about the modes of SSH client strict host key checking, their behaviors, and their effect on host key storage, automation, and system security.

- All SSH client users share the default known hosts file at `/mnt/rdsfs/ciscossh/known_hosts`. Adding, deleting, or accepting a host key in this file affects all SSH client sessions.
- RDSFS must be available to retain host keys across device reloads and switchovers.
- When you use the `ssh-client knownhosts import <file-path>` command, the imported file replaces the contents of the default known-hosts file. The system does not append the imported entries.
- Host keys learned through SSH, SCP, SFTP, or other CLI operations that initiate SSH, SCP, or SFTP sessions are stored only in the default system-wide knownhosts file.
- The following strict host key checking modes are available:
  - `strict-check off`: Automatically accepts new or changed host keys. This mode weakens protection against man-in-the-middle attacks and should be used only for temporary, low-risk troubleshooting.
  - `strict-check accept-new`: Automatically accepts and stores new host keys, but rejects changed host keys.
  - `strict-check ask`: Prompts for approval only when an unknown host key is encountered. After the key is accepted and stored, any later change to that host key is rejected.
  - `strict-check yes`: Rejects both unknown and changed host keys. To avoid connection failures, pre-populate the trusted host list before enabling this policy.



Note If the legacy `ssh client knownhost` command is present in the global configuration, you cannot configure `strict-hostkey-check`. The exec-mode `ssh-client knownhosts` commands are also unavailable until you remove the legacy configuration.

Configure SSH client strict host key checking

Configure SSH client strict host key checking to control how routers verify SSH server host keys for outbound SSH connections and prevent unauthorized access.

- Enforce trust of only verified SSH servers to meet security policies.
- Minimize operational risk by ensuring host trust decisions are policy-driven.



Note The global strict host key checking configuration can be overridden for an individual SSH, SCP, or SFTP session using the `strict-hostkey-check` option.

SSH client strict host key checking validates remote server keys before starting an SSH session. All SSH client operations reference a single, system-wide `knownhosts` file at `/mnt/rdsfs/ciscossh/known_hosts`, regardless of the user.

This mechanism ensures that clients only connect to trusted hosts and enables administrators to choose the level of enforcement (**off**, **accept-new**, **ask**, or **yes**) for their environment.

- No per-user host key management; changes are global for all SSH client users.

- The `strict-check` mode controls the degree of host verification and is set according to deployment security requirements.

Before you begin

- Administrative access to the Cisco router.
- Persistent storage is available and RDSFS is mounted at `/mnt/rdsfs`.
- Your organization's SSH security policies for trusted host validation are known.

Follow these steps to configure SSH client strict host key checking.

Procedure

Step 1 Set the SSH client strict host key checking mode.

Example:

```
Router# configure
Router(config)# ssh client strict-hostkey-check {off | accept-new | ask | yes}
Router(config)# commit
```

- `off`: Automatically accepts new and changed host keys.
- `accept-new`: Accepts and stores new host keys, but rejects changed host keys.
- `ask`: Prompts for an unknown host key; after it is accepted and stored, rejects any later key change.
- `yes`: Rejects both unknown and changed host keys.

Step 2 Exit configuration mode.

Example:

```
Router(config)# end
```

Step 3 For an individual node, use exec-mode commands to manage trusted-host entries.

The following commands operate on the current node. To align trusted-host entries across nodes, copy and import the same baseline known-hosts file on each node.

a) Import a baseline knownhosts file.

Example:

```
Router# ssh-client knownhosts import <file-path>
```

b) Export the system-wide known-hosts database.

Example:

```
Router# ssh-client knownhosts export <file-path>
```

c) Add, delete, or clear trusted-host entries if required.

Example:

```
Router# ssh-client knownhosts add
Router# ssh-client knownhosts delete hostname
Router# ssh-client knownhosts delete line
Router# ssh-client knownhosts clear
```

To manage the trusted host entries, see SSH client knownhosts commands, on page 215.

Step 4 Verify the trusted-host list.

Example:

```
Router# show ssh client knownhosts
```

SSH client strict host key checking is configured. The router handles new and changed SSH server keys according to the selected enforcement mode.

What to do next

Review the router's outbound SSH connection logs to confirm that SSH client connections are handled according to the selected strict host-key checking mode and your organization's SSH security policies.

SSH client knownhosts commands

The trusted-host database is stored in `/mnt/rdsfs/ciscossh/known_hosts` and is shared by all SSH client users.

Use these commands to manage entries in the trusted-host database.

Table 25: SSH client known hosts commands

Command	Description
ssh-client knownhosts add	Adds a trusted-host entry.
ssh-client knownhosts import	Imports a knownhosts file.
ssh-client knownhosts export	Exports the trusted-host database.
ssh-client knownhosts delete hostname	Deletes an entry by hostname.
ssh-client knownhosts delete line	Deletes a numbered entry.
ssh-client knownhosts clear	Removes all entries.
show ssh client knownhosts	Displays the trusted-host entries.

Run these commands in exec mode.

Host keys learned during SSH, SCP, SFTP, or another CLI session are stored only in the system-wide database.

Automatic generation of SSH host-key pairs

An automatic generation of SSH host-key pairs is a security feature that

- creates SSH host-key pairs for supported algorithms (DSA, ECDSA, and RSA) automatically when the router boots,
- eliminates the need for explicit manual key generation after initial setup, and
- ensures SSH clients can connect to the SSH server immediately after bootup with a basic configuration.

Table 26: Feature History Table

Feature Name	Release info	Description
SSH key strength: 3072-bit by default	Release 26.1.1	This update enhances device security by automatically generating RSA 3072-bit SSH host keys during system boot, instead of RSA 2048-bit keys. 3072-bit aligns with industry best practices and provides improved cryptographic protection, ensuring secure SSH access and compliance with the latest security requirements.

The automatic generation feature simplifies device provisioning, especially in zero touch provisioning (ZTP) and Golden ISO boot scenarios. Since SSH host-key pairs are present immediately after boot, administrators do not need to perform manual configurations to enable secure connections.

Starting Cisco IOS XR Software Release 26.1.1, the system automatically generates RSA 3072-bit SSH host keys during boot, replacing the previous default of RSA 2048-bit keys. This enhancement strengthens device security by providing improved cryptographic protection and aligns with the current industry best practices. The default RSA 3072-bit SSH host keys are generated automatically requiring no manual intervention from you.

You can choose specific algorithms to use for SSH host-key pairs by configuring the **ssh server algorithms host-key** command in Global Configuration mode. If certain key pairs are not required, use the **crypto key zeroize** command in EXEC mode to remove them. When upgrading from an earlier software version, the system automatically generates host-key pairs only if they are missing, avoiding duplicate key generation. If SSH host-key pairs are not present after bootup, you can manually generate them using the **crypto key generate** command in EXEC mode.



Note In a system upgrade scenario from version 1 to version 2, the system does not generate the SSH host-key pairs automatically if they were already generated in version 1. The host-key pairs are generated automatically only if they were not generated in version 1.

- On routers with automatic generation enabled, SSH host-key pairs for all supported algorithms are created during initial boot, enabling immediate SSH access by clients.
- During ZTP, there are no additional steps required to configure SSH host-key pairs.

Configure the Allowed SSH Host-Key Pair Algorithms

When the SSH client attempts a connection with the SSH server, it sends a list of SSH host-key pair algorithms (in the order of preference) internally in the connection request. The SSH server, in turn, picks the first matching algorithm from this request list. The server establishes a connection only if that host-key pair is already generated in the system, and if it is configured (using the **ssh server algorithms host-key** command) as the allowed algorithm.

Note If this configuration of allowed host-key pairs is not present in the SSH server, then you can consider that the SSH server allows all host-key pairs. In that case, the SSH client can connect with any one of the host-key pairs. Not having this configuration also ensures backward compatibility in system upgrade scenarios.

Configuration Example

You may perform this (optional) task to specify the allowed SSH host-key pair algorithm (in this example, **ecdsa**) from the list of auto-generated host-key pairs on the SSH server:

```
/* Example to select the ecdsa algorithm */
Router(config)#ssh server algorithms host-key ecdsa-nistp521
```

Similarly, you may configure other algorithms.

Running Configuration

```
ssh server algorithms host-key ecdsa-nistp521
!
```

Verify the SSH Host-Key Pair Algorithms

Note With the introduction of the automatic generation of SSH host-key pairs, the output of the **show crypto key mypubkey** command displays key information of all the keys that are auto-generated. Before its introduction, the output of this show command displayed key information of only those keys that you explicitly generated using the **crypto key generate** command.

```
Router#show crypto key mypubkey ecdsa
Mon Nov 19 12:22:51.762 UTC
Key label: the_default
Type     : ECDSA General Curve Nistp256
Degree   : 256
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04AC7533 3ABE7874 43F024C1 9C24CC66 490E83BE 76CEF4E2 51BBEF11 170CDB26
14289D03 6625FC4F 3E7F8F45 0DA730C3 31E960FE CF511A05 2B0AA63E 9C022482
6E

Key label: the_default
Type     : ECDSA General Curve Nistp384
Degree   : 384
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04B70BAF C096E2CA D848EE72 6562F3CC 9F12FA40 BE09BFE6 AF0CA179 F29F6407
FEE24A43 84C5A5DE D7912208 CB67EE41 58CB9640 05E9421F 2DCDC41C EED31288
6CACC8DD 861DC887 98E535C4 893CB19F 5ED3F6BC 2C90C39B 10EAED57 87E96F78
B6

Key label: the_default
Type     : ECDSA General Curve Nistp521
Degree   : 521
Created  : 10:59:09 UTC Mon Nov 19 2018
```

```
Data     :
0400BA39 E3B35E13 810D8AE5 260B8047 84E8087B 5137319A C2865629 8455928F
D3D9CE39 00E097FF 6CA369C3 EE63BA57 A4C49C02 B408F682 C2153B7F AAE53EF8
A2926001 EF113896 5F1DA056 2D62F292 B860FDFB 0314CE72 F87AA2C9 D5DD29F4
DA85AE4D 1CA453AC 412E911A 419E9B43 0A13DAD3 7B7E88E4 7D96794B 369D6247
E3DA7B8A 5E
```

The following example shows the output for **ed25519**:

```
Router#show crypto key mypubkey ed25519
Wed Dec 16 16:12:21.464 IST
Key label: the_default
Type     : ED25519
Size     : 256
Created  : 15:08:28 IST Tue Oct 13 2020
Data     :
 649CC355 40F85479 AE9BE26F B5B59153 78D171B6 F40AA53D B2E48382 BA30E5A9

Router#
```

Related Topics

#unique_244

Associated Commands

- **ssh server algorithms host-key**
- **show crypto key mypubkey**

Ed25519 Public-Key Signature Algorithm Support for SSH

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
Ed25519 Public-Key Signature Algorithm Support for SSH	Release 7.3.1	This algorithm is now supported on Cisco IOS XR 64-bit platforms when establishing SSH sessions. It is a modern and secure public-key signature algorithm that provides several benefits, particularly resistance against several side-channel attacks. Prior to this release, DSA, ECDSA, and RSA public-key algorithms were supported. This command is modified for this feature: ssh server algorithms host-key

This feature introduces the support for Ed25519 public-key algorithm, when establishing SSH sessions, on Cisco IOS XR 64-bit platforms. This algorithm offers better security with faster performance when compared to DSA or ECDSA signature algorithms.

The order of priority of public-key algorithms during SSH negotiation between the client and the server is:

- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp521
- ssh-ed25519
- ssh-rsa
- ssh-dsa

Restrictions for ED25519 Public Key for SSH

The Ed25519 public key algorithm is not FIPS-certified. That is, if FIPS mode is enabled on the router, the list of public-key algorithms sent during the SSH key negotiation phase does not contain the Ed25519 key. This behavior is applicable only for new SSH connections. Any existing SSH session that has already negotiated Ed25519 public-key algorithm remains intact and continues to execute until the session is disconnected.

Further, if you have configured the router to negotiate only the Ed25519 public-key algorithm (using the **ssh server algorithms host-key** command), and if FIPS mode is also enabled, then the SSH connection to the router fails.

How to Generate Ed25519 Public Key for SSH

To generate Ed25519 public key for SSH, see .

You must also specify Ed25519 as the permitted SSH host-key pair algorithm from the list of auto-generated host-key pairs on the SSH server. For details, see .

To remove the Ed25519 key from the router, use the **crypto key zeroize ed25519** command in XR EXEC mode.

Configure SSH Client

Perform this task to configure an SSH client.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ssh client knownhost** *device* **:** */filename*

Example:

```
RP/0/RP0/CPU0:router(config)# ssh client knownhost slot1:/server_pubkey
```

(Optional) Enables the feature to authenticate and check the server public key (pubkey) at the client end.

Note
The complete path of the filename is required. The colon (:) and slash mark (/) are also required.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 4 **ssh** {*ipv4-address* | *ipv6-address* | *hostname*} [**username** *user-* **cipher** | **source-interface** *type instance*]

Enables an outbound SSH connection.

- To run an SSHv2 server, you must have a VRF. This may be the default or a specific VRF. VRF changes are applicable only to the SSH v2 server.
- The SSH client tries to make an SSHv2 connection to the remote peer. If the remote peer supports only the SSHv1 server, the peer internally spawns an SSHv1 connection to the remote server.
- The **cipher des** option can be used only with an SSHv1 client.
- The SSHv1 client supports only the 3DES encryption algorithm option, which is still available by default for those SSH clients only.
- If the *hostname* argument is used and the host has both IPv4 and IPv6 addresses, the IPv6 address is used.

- If you are using SSHv1 and your SSH connection is being rejected, the reason could be that the RSA key pair might have been zeroed out. Another reason could be that the SSH server to which the user is connecting to using SSHv1 client does not accept SSHv1 connections. Make sure that you have specified a hostname and domain. Then use the **crypto key generate rsa** command to generate an RSA host-key pair, and then enable the SSH server.
- If you are using SSHv2 and your SSH connection is being rejected, the reason could be that the DSA, RSA host-key pair might have been zeroed out. Make sure you follow similar steps as mentioned above to generate the required host-key pairs, and then enable the SSH server.
- When configuring the ECDSA, RSA or DSA key pair, you might encounter the following error messages:
  - No hostname specified

  You must configure a hostname for the router using the **hostname** command.

  - No domain specified

You must configure a host domain for the router using the **domain-name** command.

- The number of allowable SSH connections is limited to the maximum number of virtual terminal lines configured for the router. Each SSH connection uses a vty resource.
- SSH uses either local security or the security protocol that is configured through AAA on your router for user authentication. When configuring AAA, you must ensure that the console is not running under AAA by applying a keyword in the global configuration mode to disable AAA on the console.

Note If you are using Putty version 0.63 or higher to connect to the SSH client, set the 'Chokes on PuTTYs SSH2 winadj request' option under SSH > Bugs in your Putty configuration to 'On.' This helps avoid a possible breakdown of the session whenever some long output is sent from IOS XR to the Putty client.

Configuring Secure Shell

The following example shows how to configure SSHv2 by creating a hostname, defining a domain name, enabling the SSH server for local and remote authentication on the router by generating a DSA key pair, bringing up the SSH server, and saving the configuration commands to the running configuration file.

After SSH has been configured, the SFTP feature is available on the router.

From Cisco IOS XR Software Release 7.0.1 and later, the crypto keys are auto-generated at the time of router boot up. Hence, you need to explicitly generate the host-key pair only if it is not present in the router under some scenarios.

```
configure
hostname router1
domain name cisco.com
exit
crypto key generate rsa/dsa
configure
ssh server
end
```

Order of SSH Client Authentication Methods

The default order of authentication methods for SSH clients on Cisco IOS XR routers is as follows:

- On routers running Cisco IOS XR SSH:
  - **public-key**, **password** and **keyboard-interactive** (prior to Cisco IOS XR Software Release 24.1.1)
  - **public-key**, **keyboard-interactive** and **password** (from Cisco IOS XR Software Release 24.1.1 and later)
- On routers running CiscoSSH (open source-based SSH):
  - **public-key**, **keyboard-interactive** and **password**

How to Set the Order of Authentication Methods for SSH Clients

To set the preferred order of authentication methods for SSH clients on Cisco IOS XR routers, use the **ssh client auth-method** command in the XR Config mode. This command is available from Cisco IOS XR Software Release 7.9.2/Release 7.10.1and later.

Configuration Example

In this example, we set the order of SSH client authentication methods in such a way that public key authentication is negotiated first, followed by keyboard-interactive, and then password-based authentication.

```
Router#configure
Router(config)#ssh client auth-method public-key keyboard-interactive password
Router(config-ssh)#commit
```

Running Configuration

```
Router#show run ssh client auth-methods
Tue Nov 21 17:55:44.688 IST
ssh client auth-methods public-key keyboard-interactive password
Router#
```

SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm

The Cisco IOS XR software provides a new configuration option to control the key algorithms to be negotiated with the peer while establishing an SSH connection with the router. With this feature, you can enable the insecure SSH algorithms on the SSH server, which are otherwise disabled by default. A new configuration option is also available to restrict the SSH client from choosing the HMAC, or hash-based message authentication codes algorithm while trying to connect to the SSH server on the router.

You can also configure a list of ciphers as the default cipher list, thereby having the flexibility to enable or disable any particular cipher.



Caution Use caution in enabling the insecure SSH algorithms to avoid any possible security attack.

To disable the HMAC algorithm, use the **ssh client disable hmac** command or the **ssh server disable hmac** command in XR Config mode.

To enable the required cipher, use the **ssh client enable cipher** command or the **ssh server enable cipher** command in XR Config mode.

The supported encryption algorithms (in the order of preference) are:

1. aes128-ctr
2. aes192-ctr
3. aes256-ctr
4. aes128-gcm@openssh.com
5. aes256-gcm@openssh.com

6. aes128-cbc
7. aes192-cbc
8. aes256-cbc
9. 3des-cbc

In SSH, the CBC-based ciphers are disabled by default. To enable these, you can use the **ssh client enable cipher** command or the **ssh server enable cipher** command with the respective CBC options (aes-cbc or 3des-cbc). All CTR-based and GCM-based ciphers are enabled by default.

Disable HMAC Algorithm

Configuration Example to Disable HMAC Algorithm

```
Router(config)# ssh server disable hmac hmac-sha1
Router(config)#commit
```

```
Router(config)# ssh client disable hmac hmac-sha1
Router(config)#commit
```

Running Configuration

```
ssh server disable hmac hmac-sha1
!
```

```
ssh client disable hmac hmac-sha1
!
```

Related Topics

SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 222

Associated Commands

- **ssh client disable hmac**
- **ssh server disable hmac**

Enable Cipher Public Key

Configuration Example to Enable Cipher Public Key

To enable all ciphers on the client and the server:

Router 1:

```
Router(config)# ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc
aes128-ctr aes128-cbc aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
```

Router 2:

```
Router(config)# ssh server algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc
aes128-ctr aes128-cbc aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
```

To enable the CTR cipher on the client and the CBC cipher on the server:

Router 1:

```
Router(config)# ssh client algorithms cipher aes128-ctr aes192-ctr aes256-ctr
```

Router 2:

```
Router(config)# ssh server algorithms cipher aes128-cbc aes256-cbc aes192-cbc 3des-cbc
```

Without any cipher on the client and the server:

Router 1:

```
Router(config)# no ssh client  algorithms  cipher
```

Router 2:

```
Router(config)# no ssh server  algorithms  cipher
```

Enable only the deprecated algorithms on the client and the server:

Router 1:

```
Router(config)# ssh client algorithms cipher aes128-cbc aes192-cbc aes256-cbc 3des-cbc
```

Router 2:

```
Router(config)# ssh server algorithms cipher aes128-cbc aes192-cbc aes256-cbc 3des-cbc
```

Enable the deprecated algorithm (using **enable cipher** command) and enable the CTR cipher (using **algorithms cipher** command) on the client and the server:

Router 1:

```
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
Router(config)# ssh client algorithms cipher aes128-ctr aes192-ctr aes256-ctr
```

Router 2:

```
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
Router(config)# ssh server algorithms cipher aes128-ctr aes192-ctr aes256-ctr
```

Running Configuration

All ciphers enabled on the client and the server:

Router 1:

```
ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc aes128-ctr aes128-cbc
 aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
!
```

Router 2:

```
ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc aes128-ctr aes128-cbc
 aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
!
```

Related Topics

SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 222

Associated Commands

- **ssh client enable cipher**
- **ssh server enable cipher**
- **ssh client algorithms cipher**
- **ssh server algorithms cipher**

User Configurable Maximum Authentication Attempts for SSH

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
User Configurable Maximum Authentication Attempts for SSH	Release 7.3.1	This feature allows you to set a limit on the number of user authentication attempts allowed for SSH connection, using the three authentication methods that are supported by Cisco IOS XR. The limit that you set is an overall limit that covers all the authentication methods together. If the user fails to enter the correct login credentials within the configured number of attempts, the connection is denied and the session is terminated. This command is introduced for this feature: ssh server max-auth-limit

The three SSH authentication methods that are supported by Cisco IOS XR are public-key (which includes certificate-based authentication), keyboard-interactive, and password authentication. The limit count that you set as part of this feature comes into effect whichever combination of authentication methods you use. The limit ranges from 3 to 20; default being 20 (prior to Cisco IOS XR Software Release 7.3.2, the limit range was from 4 to 20).

Restrictions for Configuring Maximum Authentication Attempts for SSH

These restrictions apply to configuring maximum authentication attempts for SSH:

- This feature is available only for Cisco IOS XR routers functioning as SSH server; not for the ones functioning as SSH clients.
- This configuration is not user-specific; the limit remains same for all the users.
- Due to security reasons, the SSH server limits the number of authentication attempts that explicitly uses the password authentication method to a maximum of 3. You cannot change this particular limit of 3 by configuring the maximum authentication attempts limit for SSH.

  For example, even if you configure the maximum authentication attempts limit as 5, the number of authentication attempts allowed using the password authentication method still remain as 3.

Configure Maximum Authentication Attempts for SSH

You can use the **ssh server max-auth-limit** command to specify the maximum number of authentication attempts allowed for SSH connection.

Configuration Example

```
Router#configure
Router(config)#ssh server max-auth-limit 5
Router(config)#commit
```

Running Configuration

```
Router#show running-configuration ssh
ssh server max-auth-limit 5
ssh server v2
!
```

Verification

The system displays the following SYSLOG on the router console when maximum authentication attempts is reached:

```
RP/0/RP0/CPU0:Oct 6 10:03:58.029 UTC: SSHD_[68125]: %SECURITY-SSHD-3-ERR_GENERAL : Max
authentication tries reached-exiting
```

Associated Commands

- **ssh server max-auth-limit**

X.509v3 Certificate-based Authentication for SSH

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
X.509v3 Certificate-based Authentication for SSH	Release 7.3.1	This feature adds new public-key algorithms that use X.509v3 digital certificates for SSH authentication. These certificates use a chain of signatures by a trusted certification authority to bind a public key to the digital identity of the user who is authenticating with the SSH server. These certificates are difficult to falsify and therefore used for identity management and access control across many applications and networks. Commands introduced for this feature are: ssh server certificate ssh server trustpoint This command is modified for this feature: ssh server algorithms host-key

This feature adds new public-key algorithms that use X.509v3 digital certificates for SSH authentication. This feature support is available for the SSH server for server and user authentication.

The X.509v3 certificate-based authentication for SSH feature supports the following public-key algorithms:

- **x509v3-ssh-dss**
- **x509v3-ssh-rsa**
- **x509v3-ecdsa-sha2-nistp256**
- **x509v3-ecdsa-sha2-nistp384**
- **x509v3-ecdsa-sha2-nistp521**



Note While user authentication by using X.509v3 certificate-based authentication for the SSH server is supported using all algorithms listed above, server authentication is supported only with the **x509v3-ssh-rsa** algorithm.

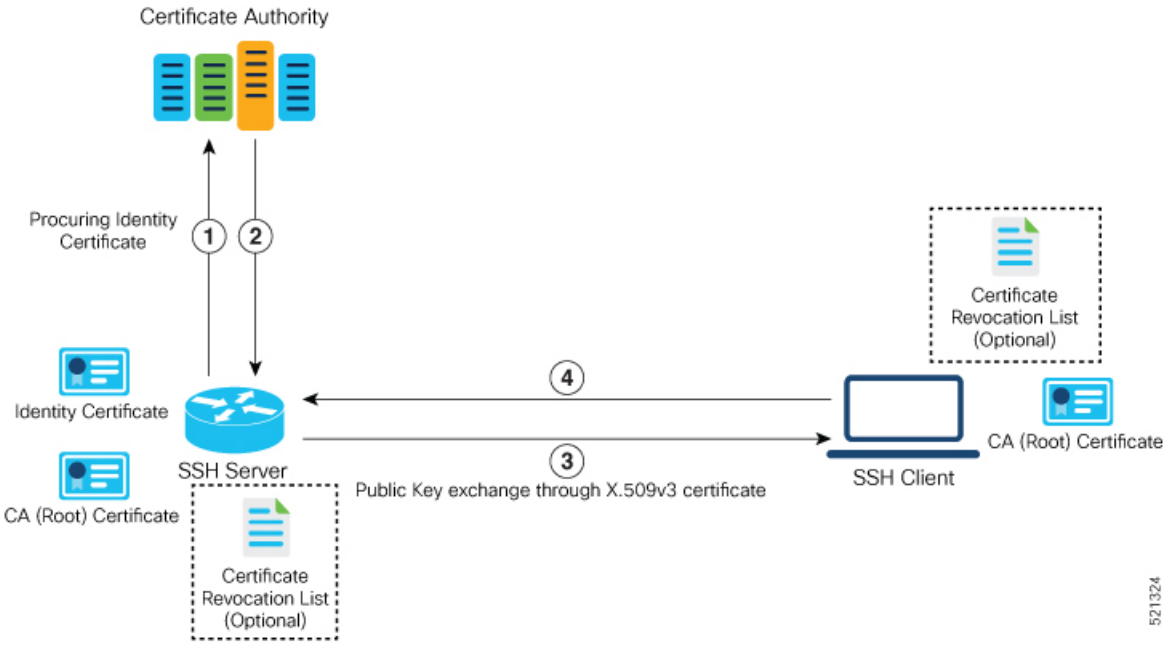
There are two SSH protocols that use public-key cryptography for authentication:

- Transport Layer Protocol (TLP) described in RFC4253—this protocol mandates that you use a digital signature algorithm (called the public-key algorithm) to authenticate the server to the client.

- User Authentication Protocol (UAP) described in RFC4252—this protocol allows the use of a digital signature to authenticate the client to the server (public-key authentication).

For TLP, the Cisco IOS XR SSH server provides its server certificate to the client, and the client verifies the certificate. Similarly, for UAP, the client provides an X.509 certificate to the server. The peer checks the validity and revocation status of the certificate. Based on the result, access is allowed or denied.

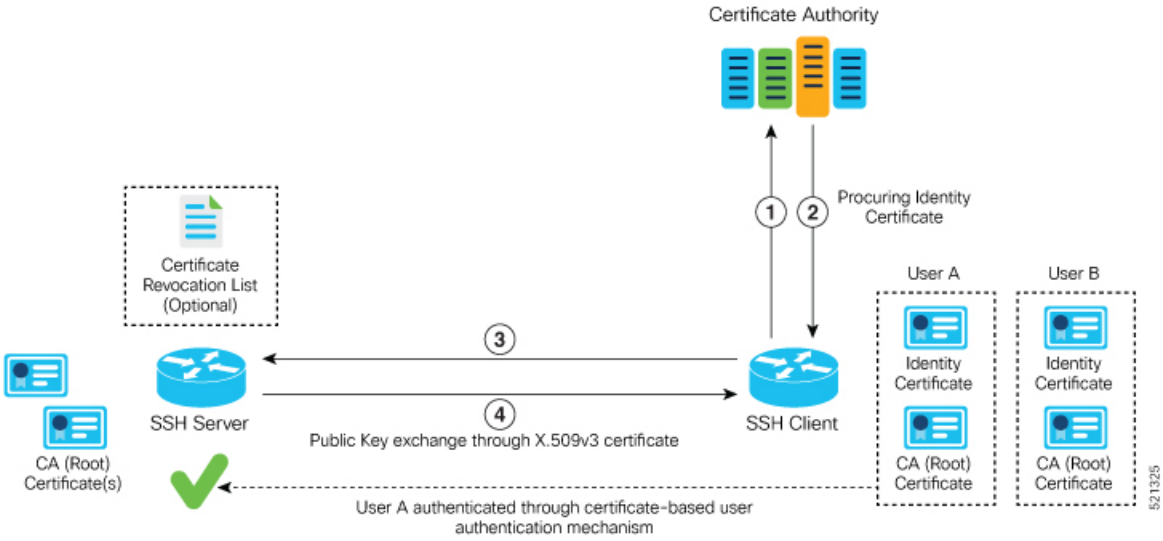
Server Authentication using X.509v3 Certificate



The server authentication process involves these steps:

1. The SSH server procures a valid identity certificate from a well-known certificate authority. This certificate can be obtained manually (through cut-and-paste mechanism) or through protocol implementations such as Simple Certificate Enrollment Protocol (SCEP).
2. The certificate authority provides valid identity certificates and associated root certificates. The requesting device stores these certificates locally.
3. The SSH server presents the certificate to the SSH client for verification.
4. The SSH client validates the certificate and starts the next phase of the SSH connection.

User Authentication using X.509v3 Certificate



The user authentication phase starts after the SSH transport layer is established. At the beginning of this phase, the client sends the user authentication request to the SSH server with required parameters. The user authentication process involves these steps:

1. The SSH client requests a valid identity certificate from a well-known certificate authority.
2. The certificate authority provides valid identity certificates and associated root certificates. The requesting device stores these certificates locally.
3. The SSH client presents the certificate to the SSH server for verification.
4. The SSH server validates the certificate and starts the next phase of the SSH connection.

The certificate-based authentication uses public key as the authentication method. The certificate validation process by the SSH server involves these steps:

- The SSH server retrieves the user authentication parameters, verifies the certificate, and also checks for the certificate revocation list (CRL).
- The SSH server extracts the *username* from the certificate attributes, such as *subject name* or *subject alternate name* (SAN) and presents them to the AAA server for authorization.
- The SSH server then takes the extracted *username* and validates it against the incoming *username* string present in the SSH connection parameter list.

Restrictions for X.509v3 Certificate-based Authentication for SSH

These restrictions apply to the X.509v3 certificate-based authentication feature for SSH:

- Supported only for Cisco IOS XR devices acting as the SSH server; not for the Cisco IOS XR devices acting as the SSH client.
- Supported only for local users because TACACS and RADIUS server do not support public-key authentication. As a result, you must include the **local** option for AAA authentication configuration.

Note Although this feature supports only local authentication, you can enforce remote authorization and accounting using the TACACS server.

- Certificate verification using the Online Certificate Status Protocol (OCSP) is currently not supported. The revocation status of certificates is checked using a certificate revocation list (CRL).
- To avoid user authentication failure, the chain length of the user certificate must not exceed the maximum limit of 9.

Configure X.509v3 Certificate-based Authentication for SSH

To enable X.509v3 certificate-based authentication for SSH, these tasks for server and user authentication:

Server Authentication:

- Configure the list of host key algorithms—With this configuration, the SSH server decides the list of host keys to be offered to the client. In the absence of this configuration, the SSH server sends all available algorithms to the user as host key algorithms. The SSH server sends these algorithms based on the availability of the key or the certificate.
- Configure the SSH trust point for server authentication—With this configuration, the SSH server uses the given trust point certificate for server authentication. In the absence of this configuration, the SSH server does not send **x509v3-ssh-rsa** as a method for server verification. This configuration is not VRF-specific; it is applicable to SSH running in all VRFs.

  The above two tasks are for server authentication and the following ones are for user authentication.

User Authentication:

- Configure the trust points for user authentication—With this configuration, the SSH server uses the given trust point for user authentication. This configuration is not user-specific; the configured trust points are used for all users. In the absence of this configuration, the SSH server does not authenticate using certificates. This configuration is not specific to a VRF; it is applicable to SSH running in all VRFs.

  You can configure up to ten user trust points.
- Specify the *username* to be picked up from the certificate—This configuration specifies which field in the certificate is to be considered as the *username*. The **common-name** from the **subject name** or the **user-principle-name(othername)** from the **subject alternate name**, or both can be configured.
- Specify the maximum number of authentication attempts allowed by the SSH server—The value ranges from 4 to 20. The default value is 20. The server closes the connection if the number of user attempts exceed the configured value.
- AAA authentication configuration—The AAA configuration for public key is the same as that for the regular or keyboard-interactive authentication, except that it mandates local method in the authentication method list.

Configuration Example

In this example, the **x509v3-ssh-rsa** is specified as the allowed host key algorithm to be sent to the client. Similarly, you can configure other algorithms, such as **ecdsa-sha2-nistp521, ecdsa-sha2-nistp384, ecdsa-sha2-nistp256, ssh-rsa**, and **ssh-dsa**.

```
/* Configure the lits of host key algorithms */
Router#configure
Router(config)#ssh server algorithms host-key x509v3-ssh-rsa
Router(config)#commit

/* Configure the SSH trustpoint for server authentication */
Router#configure
Router(config)#ssh server certificate trustpoint host tp1
Router(config)#commit

/* Configure the trustpoints to be used for user authentication */
Router#configure
Router(config)#ssh server trustpoint user tp1
Router(config)#ssh server trustpoint user tp2
Router(config)#commit

/* Specifies the username to be picked up from the certificate.
In this example, it specifies the user common name to be picked up from the subject name
field */
Router#configure
Router(config)#ssh server certificate username common-name
Router(config)#commit

/* Specifies the maximum authentication limit for the SSH server */
Router#configure
Router(config)#ssh server max-auth-limit 5
Router(config)#commit

/* AAA configuration for local authentication with certificate and
remote authorization with TACACS+ or RADIUS */
Router#configure
Router(config)#aaa authentication login default group tacacs+ local
Router(config)#aaa authorization exec default group radius group tacacs+
Router(config)#commit
```

Running Configuration

```
ssh server algorithms host-key x509v3-ssh-rsa
!
ssh server certificate trustpoint host tp1
!
ssh server trustpoint user tp1
ssh server trustpoint user tp2
!
ssh server certificate username common-name
!
ssh server max-auth-limit 5
!
```

Verification of Certificate-based Authentication for SSH

You can use the **show ssh server** command to see various parameters of the SSH server. For certificate-based authentication for SSH, the **Certificate Based** field displays *Yes*. Also, the two new fields, **Host Trustpoint** and **User Trustpoints**, display the respective trust point names.

```
Router#show ssh server
Wed Feb 19 15:23:38.752 IST
---------------------
SSH Server Parameters
---------------------

Current supported versions := v2
                  SSH port := 22
                  SSH vrfs := vrfname:=default(v4-acl:=, v6-acl:=)
              Netconf Port := 830
              Netconf Vrfs := vrfname:=default(v4-acl:=, v6-acl:=)

 Algorithms
---------------
        Hostkey Algorithms := x509v3-ssh-rsa,
ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,ssh-rsa,ssh-dsa
   Key-Exchange Algorithms :=
ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group14-sha1
     Encryption Algorithms :=
aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
            Mac Algorithms := hmac-sha2-512,hmac-sha2-256,hmac-sha1

 Authetication Method Supported
-----------------------------------
                 PublicKey := Yes
                  Password := Yes
      Keyboard-Interactive := Yes
         Certificate Based := Yes

 Others
------------
                     DSCP  := 16
                Ratelimit  := 60
             Sessionlimit  := 100
                Rekeytime  := 60
       Server rekeyvolume  := 1024
  TCP window scale factor  := 1
            Backup Server  := Enabled, vrf:=default, port:=11000
Host Trustpoint            := tp1
User Trustpoints           := tp1 tp2
```

You can use the **show ssh session details** command to see the chosen algorithm for an SSH session:

```
Router#show ssh session details
Wed Feb 19 15:33:00.405 IST
SSH version : Cisco-2.0

id      key-exchange           pubkey               incipher    outcipher   inmac
outmac
-------------------------------------------------------------------------------------------------

Incoming Sessions
1       ecdh-sha2-nistp256     x509v3-ssh-rsa       aes128-ctr  aes128-ctr  hmac-sha2-256
hmac-sha2-256
```

Similarly, you can use the **show ssh** command to verify the authentication method used. In this example, it shows as *x509-rsa-pubkey*:

```
Router#show ssh
Sun Sep 20 18:14:04.122 UTC
SSH version : Cisco-2.0

id  chan pty  location   state        userid    host          ver authentication connection
 type
-----------------------------------------------------------------------------------------------
Incoming sessions
4   1    vty0 0/RP0/CPU0 SESSION_OPEN  9chainuser 10.105.230.198 v2  x509-rsa-pubkey
Command-Line-Interface

Outgoing sessions
```

SYSLOGS

You can observe relevant SYSLOGS on the router console in various scenarios listed here:

- On successful verification of peer certificate:

```
RP/0/RP0/CPU0:Aug 10 15:01:34.793 UTC: locald_DLRSC[133]: %SECURITY-PKI-6-LOG_INFO :
Peer certificate verified successfully
```

- When user certificate CA is not found in the trust point:

```
RP/0/RP0/CPU0:Aug  9 22:06:43.714 UTC: locald_DLRSC[260]: %SECURITY-PKI-3-ERR_GENERAL
: issuer not found in trustpoints configured
RP/0/RP0/CPU0:Aug  9 22:06:43.714 UTC: locald_DLRSC[260]: %SECURITY-PKI-3-ERR_ERRNO :
Error:='Crypto Engine' detected the 'warning' condition 'Invalid trustpoint or trustpoint
 not exist'(0x4214c000), cert verificationn failed
```

- When there is no CA certificate or host certificate in the trust point:

```
RP/0/RP1/CPU0:Aug 10 00:23:28.053 IST: SSHD_[69552]: %SECURITY-SSHD-4-WARNING_X509 :
could not get the host cert chain, 'sysdb' detected the 'warning' condition 'A SysDB
client tried to access a nonexistent item or list an empty directory', x509 host auth
will not be used
RP/0/RP1/CPU0:Aug 10 00:23:30.442 IST: locald_DLRSC[326]: %SECURITY-PKI-3-ERR_ERRNO :
Error:='Crypto Engine' detected the 'warning' condition 'Invalid trustpoint or trustpoint
 not exist'(0x4214c000), Failed to get trustpoint name from
```

How to Disable X.509v3 Certificate-based Authentication for SSH

- Server Authentication — You can disable X.509v3 certificate-based server authentication for SSH by using the **ssh server algorithms host-key** command. From the list of auto-generated host-key pairs algorithms on the SSH server, this command configures allowed SSH host-key pair algorithms. Hence, if you have this configuration without specifying the **x509-ssh-rsa** option in the preceding command, it is equivalent to disabling the X.509v3 certificate-based server authentication for the SSH server.
- User Authentication — You can remove the user trust point configuration (**ssh server trustpoint user**) so that the SSH server does not allow the X.509v3 certificate-based authentication.

Failure Modes for X.509v3 Certificate-based Authentication for SSH

If the **ssh server certificate trustpoint host** configuration is missing, or if the configuration is present, but the router certificate is not present under the trust point, then the SSH server does not add **x509-ssh-rsa** to the list of supported host key methods during key exchange.

Also, the user authentication fails with an error message if:

- User certificate is in an incorrect format.
- The chain length of the user certificate is more than the maximum limit of 9.
- Certificate verification fails due to any reason.

Related Topics

- X.509v3 Certificate-based Authentication for SSH, on page 227

Associated Commands

- **ssh server algorithms hostkey**
- **ssh server certificate username**
- **ssh server max-auth-limit**
- **ssh server trustpoint host**
- **ssh server trustpoint user**
- **show ssh server**
- **show ssh session details**

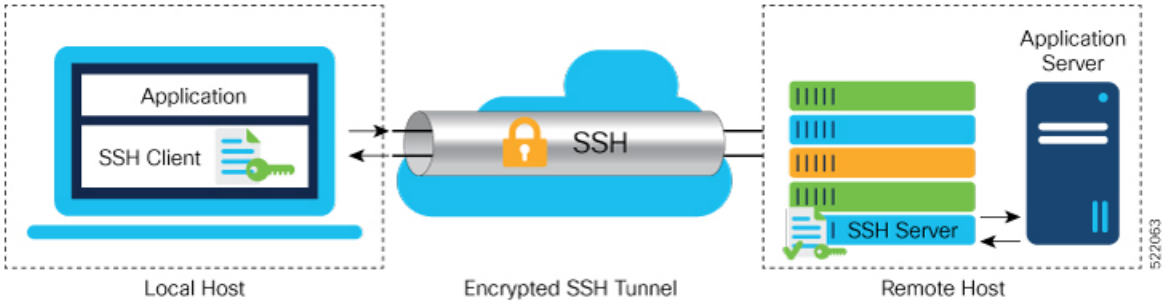
SSH Port Forwarding

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
SSH Port Forwarding	Release 7.3.2	With this feature enabled, the SSH client on a local host forwards the traffic coming on a given port to the specified host and port on a remote server, through an encrypted SSH channel. Legacy applications that do not otherwise support data encryption can leverage this functionality to ensure network security and confidentiality to the traffic that is sent to remote application servers. This feature introduces the ssh server port-forwarding local command.

SSH port forwarding is a method of forwarding the otherwise insecure TCP/IP connections from the SSH client to server through a secure SSH channel. Since the traffic is directed to flow through an encrypted SSH connection, it is tough to snoop or intercept this traffic while in transit. This SSH tunneling provides network security and confidentiality to the data traffic, and hence legacy applications that do not otherwise support encryption can mainly benefit out of this feature. You can also use this feature to implement VPN and to access intranet services across firewalls.

Figure 5: SSH Port Forwarding

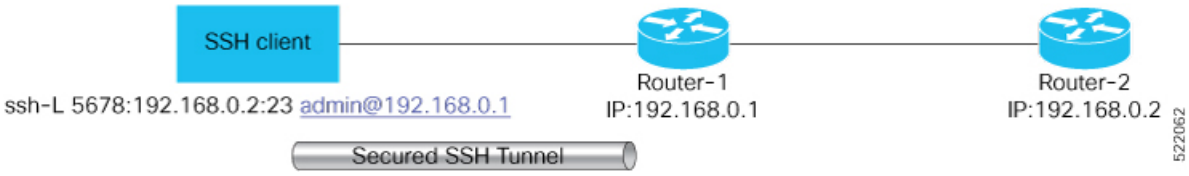


Consider an application on the SSH client residing on a local host, trying to connect to an application server residing on a remote host. With tunneling enabled, the application on the SSH client connects to a port on the local host that the SSH client listens to. The SSH client then forwards the data traffic of the application to the SSH server over an encrypted tunnel. The SSH server then connects to the actual application server that is either residing on the same router or on the same data center as the SSH server. The entire communication of the application is thus secured, without having to modify the application or the work flow of the end user.

The SSH port forwarding feature is disabled, by default. You can enable the feature by using the **ssh server port-forwarding local** command in the XR Config mode.

How Does SSH Port Forwarding Work?

Figure 6: Sample Topology for SSH Port Forwarding



Consider a scenario where port forwarding is enabled on the SSH server running on Router-1, in this topology. An SSH client running on a local host tries to create a secure tunnel to the SSH server, for a local application to eventually reach the remote application server running on Router-2.

The client tries to establish an SSH connection to Router-1 using the following command:

```
ssh -L local-port:remote-server-hostname:remote-port username@sshserver-hostname
```

where,

local-port is the local port number of the host where the SSH client and the application reside. Port 5678, in this example.

remote-server-hostname:remote-port is the TCP/IP host name and port number of the remote application server where the recipient (SSH server) must connect the channel from the SSH client to. 192.168.0.2 and 23, in this example.

sshserver-hostname is the domain name or IP address of the SSH server which is the recipient of the SSH client request. 192.168.0.1, in this example.

For example,

```
ssh -L 5678:192.168.0.2:23 admin@192.168.0.1
```

When the SSH server receives a TCP/IP packet from the SSH client, it accepts the packet and opens a socket to the remote server and port specified in that packet. Once the connection between SSH client and server is established, the SSH server connects that communication channel to the newly created socket. From then onwards, SSH server forwards all the incoming data from the client on that channel to that socket. This type of connection is known as port-forwarded local connection. When the client closes the connection, the SSH server closes the socket and the forwarded channel.

How to Enable SSH Port Forwarding

Guidelines for Enabling SSH Port Forwarding Feature

- The Cisco IOS XR software supports SSH port forwarding only on SSH server; not on SSH client. Hence, to utilize this feature, the SSH client running at the end host must already have the support for SSH port forwarding or tunneling.
- The remote host must be reachable on the same VRF where the current SSH connection between the server and the client is established.
- Port numbers need not need match for SSH port forwarding to work. You can map any port on the SSH server to any port on the client.

- If the SSH client tries to do port forwarding without the feature being enabled on the SSH server, the port forwarding fails, and displays an error message on the console. Similarly the port-forwarded channel closes in case there is any connectivity issue or if the server receives an SSH packet from the client in an improper format.

Configuration Example

```
Router#configure
Router(config)#ssh server port-forwarding local
Router(config)#commit
```

Running Configuration

```
Router#show running-configuration

ssh server port-forwarding local
!
```

Verification

Use the **show ssh** command to see the details of the SSH sessions. The **connection type** field shows as **tcp-forwarded-local** for the port-forwarded session.

```
Router#show ssh

Wed Oct 14 11:22:05.575 UTC
SSH version : Cisco-2.0

id chan pty  location   state        userid host          ver authentication connection
type
-----------------------------------------------------------------------------------------
Incoming sessions
15 1    XXX  0/RP0/CPU0 SESSION_OPEN  admin  192.168.122.1 v2  password
port-forwarded-local

Outgoing sessions

Router#
```

Use the **show ssh server** command to see the details of the SSH server. The **Port Forwarding** column shows as **local** for the port-forwarded session. Whereas, for a regular SSH session, the field displays as **disabled**.

```
Router#show ssh server
```

Syslogs for SSH Port Forwarding Feature

The router console displays the following syslogs at various SSH session establishment events.

- When SSH port forwarding session is successfully established:

```
RP/0/RP0/CPU0:Aug 24 13:10:15.933 IST: SSHD_[66632]:
%SECURITY-SSHD-6-PORT_FWD_INFO_GENERAL : Port Forwarding, Target:=10.105.236.155,
Port:=22, Originator:=127.0.0.1,Port:=41590, Vrf:=0x60000000, Connection forwarded
```

- If SSH client tries to establish a port forwarding session without SSH port forwarding feature being enabled on the SSH server:

```
RP/0/RP0/CPU0:Aug 24 13:20:31.572 IST: SSHD_[65883]: %SECURITY-SSHD-3-PORT_FWD_ERR_GENERAL
 : Port Forwarding, Port forwarding is not enabled
```

Associated Command

- **ssh server port-forwarding local**

Non-Default SSH Port

Table 31: Feature History Table

Feature Name	Release Information	Feature Description
Non-Default SSH Port	Release 7.7.1	We have enhanced the system security to minimize the automated attacks that may target the default Secure Socket Shell (SSH) port on your router. You can now specify a non-default port number for the SSH server on your router. The SSH, Secure Copy Protocol (SCP), and Secure File Transfer Protocol (SFTP) client services can then access your router only through this non-default port. The new port option also enables the SSH, SCP, and SFTP clients on your router to connect to SSH servers on the network that use a wide range of non-default port numbers. In earlier releases, these SSH, SCP, and SFTP connections were established through the default SSH port, 22. The non-default SSH port is supported only on SSH version 2. The feature introduces the command. The feature modifies these commands to include the **port** option: • **ssh** • **sftp** • **scp**

The SSH, SCP, and SFTP services on the Cisco IOS XR routers used the default SSH port number, 22, to establish connections between the server and the client. From Cisco IOS XR Software Release 7.7.1 and later, you can specify a non-default SSH port number within a specific range for these services on Cisco IOS XR 64-bit routers. This non-default port option is available for routers that are functioning as servers, or as clients for the SSH, SCP and SFTP services. This feature helps to restrict insecure client services from accessing the router through the default SSH server port. Similarly, for Cisco IOS XR routers that are running as SSH clients, the non-default port number option enables them to connect to other SSH servers on the network that listens on a wide range of non-default SSH port numbers.

The non-default SSH port number ranges from 5520 to 5529 for the SSH server, and from 1025 to 65535 for the SSH client.

The SSH server on the router does not listen on both the default and non-default ports at the same time. If you have configured a non-default SSH server port, then the server listens only on that non-default port for the client connections. The SSH clients can then establish sessions through this non-default SSH port. The SCP and SFTP services also use the same SSH port for their connections, and hence they establish the client sessions through the newly configured port.

If a session was already established through the default port, then that session remains intact even if you change the ssh server port to a non-default port. The further client sessions are attempted through the newly configured non-default port.

Restrictions for Non-Default SSH Port

These restrictions apply to the non-default SSH port option:

- Available only on 64-bit Cisco IOS XR routers; not on 32-bit routers
- Available only on version 2 of SSH (SSHv2); not on version 1 (SSHv1)

How to Configure Non-Default SSH Port

Note To establish SSH connections on the non-default port, ensure that the non-default port that you select for the SSH server is not used by any other application on the router.

Configuration Example

SSH Server:

To configure the non-default SSH port for the SSH server on the router, use the **ssh server port** command in the XR Config mode.

```
Router#configure
Router(config)#ssh server port 5520
Router(config)#commit
```

SSH Client:

Similarly, the **port** option is available for the SSH client also, to initiate a connection to another SSH server that listens on a non-default SSH port number.

This example shows how to connect to an SSH server, with IP address 198.51.100.1, that is listening on non-default SSH port 5525.

```
Router#ssh 198.51.100.1 port 5525 username user1
```

Running Configuration

This is a sample running configuration of the SSH server.

```
Router#show running-configuration
!
ssh server v2
ssh server port 5520
ssh server vrf default
!
```

Verification

Use the following **show** commands to verify the SSH server configuration and LPTS entries for SSH connections.

In this example, the **SSH port** field displays the port number, '5520', that you have configured for the SSH server.

```
Router#show ssh server
Fri May 20 07:22:57.579 UTC
---------------------
SSH Server Parameters
---------------------

Current supported versions := v2
                  SSH port := 5520
                  SSH vrfs := vrfname:=default(v4-acl:=, v6-acl:=)
              Netconf Port := 830
              Netconf Vrfs :=

 Algorithms
---------------
        Hostkey Algorithms :=
x509v3-ssh-rsa,ssh-rsa-cert-v01@openssh.com,ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,rsa-sha2-512,rsa-sha2-256,ssh-rsa,ssh-dss,ssh-ed25519

   Key-Exchange Algorithms :=
ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group14-sha1,curve25519-sha256,diffie-hellman-group14-sha256,diffie-hellman-group16-sha512,curve25519-sha256@libssh.org

     Encryption Algorithms :=
aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com

            Mac Algorithms := hmac-sha2-512,hmac-sha2-256,hmac-sha1

Authentication Method Supported
------------------------------------
                 PublicKey := Yes
                  Password := Yes
      Keyboard-Interactive := Yes
         Certificate Based := Yes

 Others
------------
                     DSCP  := 16
                Ratelimit  := 60
```

```
            Sessionlimit  := 64
               Rekeytime  := 60
       Server rekeyvolume  := 1024
  TCP window scale factor  := 1
            Backup Server  := Disabled
          Host Trustpoint  :=
          User Trustpoint  :=
          Port Forwarding  := Disabled
Max Authentication Limit  := 20
     Certificate username  := Common name(CN)
  OpenSSH Host Trustpoint  :=
  OpenSSH User Trustpoint  :=
```

In the following example, the **Port** field in the **Local-Address,Port** column for the **TCP** entry for SSH displays the port number as '5520'. This is the port on which the SSH server listens for client connections.

```
Router#show lpts bindings brief
Fri May 20 07:23:21.416 UTC

@ - Indirect binding; Sc - Scope

Location   Clnt Sc L3   L4     VRF-ID    Interface    Local-Address,Port Remote-Address,Port
---------- ---- -- ---- ------ --------- ------------ ------------------------------------
0/RP0/CPU0 IPV4 LO IPV4 ICMP   *         any          any,ECHO           any
0/RP0/CPU0 IPV4 LO IPV4 ICMP   *         any          any,TSTAMP         any
0/RP0/CPU0 IPV4 LO IPV4 ICMP   *         any          any,MASKREQ        any
0/RP0/CPU0 IPV6 LO IPV6 ICMP6  *         any          any,ECHOREQ        any
0/RP0/CPU0 IPV6 LO IPV6 ICMP6  *         any          any,NDRTRSLCT      any
0/RP0/CPU0 IPV6 LO IPV6 ICMP6  *         any          any,NDRTRADV       any
0/RP0/CPU0 IPV6 LO IPV6 ICMP6  *         any          any,NDNBRSLCT      any
0/RP0/CPU0 IPV6 LO IPV6 ICMP6  *         any          any,NDNBRADV       any
0/RP0/CPU0 IPV6 LO IPV6 ICMP6  *         any          any,NDREDIRECT     any
0/RP0/CPU0 BFD  LO IPV4 UDP    *         any          any                any
0/0/CPU0   IPV4 LO IPV4 ICMP   *         any          any,ECHO           any
0/0/CPU0   IPV4 LO IPV4 ICMP   *         any          any,TSTAMP         any
0/0/CPU0   IPV4 LO IPV4 ICMP   *         any          any,MASKREQ        any
0/0/CPU0   IPV6 LO IPV6 ICMP6  *         any          any,ECHOREQ        any
0/0/CPU0   IPV6 LO IPV6 ICMP6  *         any          any,NDRTRSLCT      any
0/0/CPU0   IPV6 LO IPV6 ICMP6  *         any          any,NDRTRADV       any
0/0/CPU0   IPV6 LO IPV6 ICMP6  *         any          any,NDNBRSLCT      any
0/0/CPU0   IPV6 LO IPV6 ICMP6  *         any          any,NDNBRADV       any
0/0/CPU0   IPV6 LO IPV6 ICMP6  *         any          any,NDREDIRECT     any
0/0/CPU0   BFD  LR IPV4 UDP    *         any          any 128.64.0.0/16
0/RP0/CPU0 TCP  LR IPV6 TCP    default   any          any,5520           any
0/RP0/CPU0 TCP  LR IPV4 TCP    default   any          any,5520           any
0/RP0/CPU0 UDP  LR IPV6 UDP    default   any          any,33433          any
0/RP0/CPU0 UDP  LR IPV4 UDP    default   any          any,33433          any
0/RP0/CPU0 RAW  LR IPV4 IGMP   default   any          any                any
0/RP0/CPU0 RAW  LR IPV4 L2TPV3 default   any          any                any
0/RP0/CPU0 RAW  LR IPV6 ICMP6  default   any          any,MLDLQUERY      any
0/RP0/CPU0 RAW  LR IPV6 ICMP6  default   any          any,LSTNRREPORT    any
0/RP0/CPU0 RAW  LR IPV6 ICMP6  default   any          any,MLDLSTNRDN     any
0/RP0/CPU0 RAW  LR IPV6 ICMP6  default   any          any,LSTNRREPORT    any

Router#
```

If the non-default port was not configured, then the SSH server listens on the default SSH port 22, and the above **Port** field displays '22'.

If a session was already established through the default port, and if you change the ssh server port to a non-default port, then the output still displays an entry for that session on the default port, 22. Another entry

shows that the SSH server is listening on the newly configured non-default port. New connections establish through the non-default port, 5520, in this example.

```
Location   Clnt Sc L3   L4  VRF-ID    Interface Local-Address,Port Remote-Address,Port
---------- ---- -- ---- --- --------- --------- -----------------  ------------------
.
.
.
0/RP0/CPU0 TCP  LR IPV4 TCP default   any       192.0.2.1,5520    198.51.100.1,37764
0/RP0/CPU0 TCP  LR IPV4 TCP default   any       any,5520 any
0/RP0/CPU0 TCP  LR IPV6 TCP default   any       any,5520 any
0/RP0/CPU0 TCP  LR IPV4 TCP default   any       192.0.2.1,22      198.51.100.1,45722
.
.
.
```

Multi-Factor Authentication for SSH

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
Multi-Factor Authentication for SSH	Release 24.1.1	You can now deploy robust authentication mechanisms for SSH connections to your routers and reduce security risks due to compromised or weak passwords. We now support multi-factor authentication (MFA)—a secure access management solution that verifies the identity of a user using multiple verification factors—for SSH login on Cisco IOS XR routers. These verification factors include a combination of login credentials such as username and password and a token, a cryptographic device, or a mobile phone with MFA application installed. No new commands or data models were introduced or modified as part of this feature.

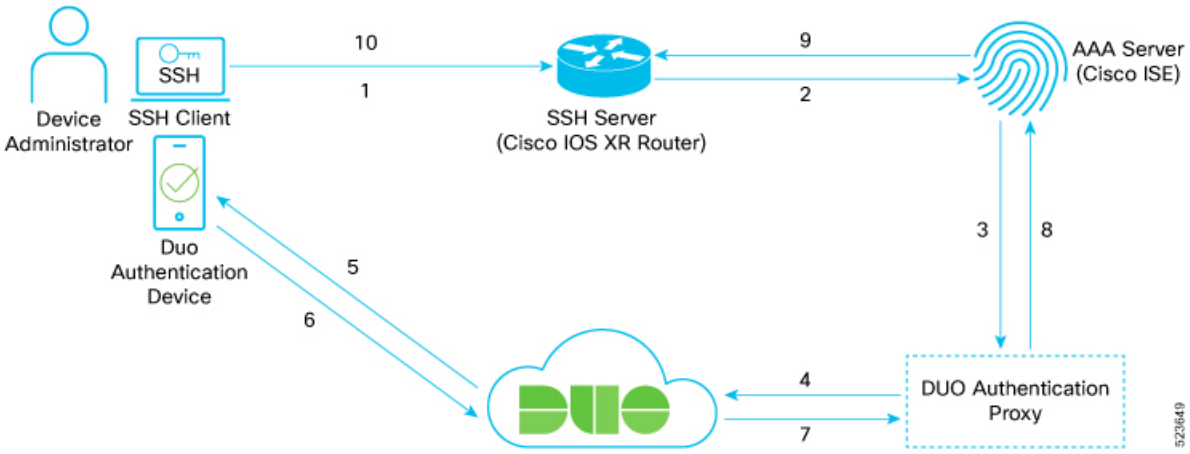
Multi-factor authentication is a multi-step authentication process that requires users to enter two or more verification factors to gain access to a system. These verification factors include something you know—such as a username and a password, and something you have—such as a token, a cryptographic authentication device, or a mobile phone with MFA application installed. MFA thereby enables stronger authentication mechanism and reduces security risk to the network devices arising due to compromised or weak passwords.

To achieve MFA for SSH, the SSH server as well as the client must support keyboard-interactive authentication method. The default order of SSH client authentication methods to support MFA in Cisco IOS XR routers is public-key, keyboard-interactive, and password-based authentication. You can change this default order as per your requirement using the **ssh client auth-method** command.

Multi-Factor Authentication Workflow

This is a sample topology to demonstrate the MFA workflow to establish SSH connection on a Cisco IOS XR router. In this example we have considered Cisco IOS XR router as the SSH server, Cisco ISE as the AAA server, and Cisco DUO authentication proxy and cloud services for MFA.

Figure 7: Multi-Factor Authentication Set-up for SSH Connection: Sample Topology



Key Components

The key components in this sample Duo MFA topology for SSH include:

- SSH client—from where the admin user initiates SSH connection to the SSH server.
- SSH server—which is the network device or router to which SSH connection is to be established.
- Cisco identity services engine (ISE)—that acts as the RADIUS or TACACS+ Server for AAA.
- DUO authentication proxy—is an on-premises software service that receives authentication requests from your local devices and applications through RADIUS or LDAP, optionally performs primary authentication against your existing LDAP directory or RADIUS authentication server, and then contacts Duo to perform secondary authentication.
- DUO cloud service—Cisco cloud-based security platform that provides secure access to any device or application.
- DUO authentication device—such as a mobile phone which has the Duo application installed.

The detailed workflow of Duo MFA for SSH is as follows:

1. The admin user initiates an SSH connection to the SSH server (Cisco IOS XR router, in this case) using the login credentials of the users that are already configured on ISE.
2. The router forwards the request to the TACACS+ AAA server (Cisco ISE, in this case).
3. The Cisco ISE sends the authentication request to Duo authentication proxy. The proxy forwards the request back to ISE for the 1st factor authentication. ISE informs the authentication proxy if the local authentication was successful.
4. Upon successful ISE authentication, the authentication proxy sends an authentication request to Duo cloud for 2nd factor authentication.

5. Duo cloud sends a *PUSH* notification to the DUO authentication device of the admin user.
6. The admin user approves the *PUSH* notification.
7. The Duo cloud informs the authentication proxy of the successful *PUSH* notification.
8. The authentication proxy informs ISE of a successful authentication.
9. The ISE authorizes the admin user.
10. The admin user successfully establishes an SSH connection with the router.

Set Up Multi-Factor Authentication for SSH

This section describes how to set up a sample topology for establishing SSH connection with Cisco IOS XR router using Duo MFA.

Prerequisites

- The Cisco IOS XR router installed with Cisco IOS XR Software Release 24.1.1 or later, that acts as the server to the SSH client, and as the client to the ISE server. The router must be already configured for AAA with ISE.
- Cisco identity services engine (ISE) server that acts as the RADIUS or TACACS+ AAA server.
- Duo MFA proxy application must be installed on either Windows or on Linux machine. For details, see https://duo.com/docs/authproxy-reference.
- DUO application must be installed on the DUO authentication device.

The procedure to set up MFA for SSH involves these high-level tasks:

- Configure Duo System
- Configure Duo Authentication Proxy
- Configure ISE
- Configure RADIUS Server Attributes on the Router
- Verify Duo MFA Set-up

Configure Duo System for MFA

Configuring Duo system for MFA involves these key steps:

1. Create a Duo account in https://duo.com/
2. Perform these Duo system configurations (for details, see the *First Steps* listed in https://duo.com/docs/radius):
   - Login to your Duo account and click on **Applications**.
   - Search for **Cisco ISE server** and click on **Protect This Application**.
   - In a notepad copy and paste your **Integration Key**, **Secret Key**, and **API Hostname**.

3. Add Duo mobile device:

   Select **Dashboard** > **Users** > ***username*** > **Add Phone**

4. Activate Duo mobile:

   Select **Dashboard** > **2FA Devices** > ***phone-number*** > **Activate Duo Mobile**

Configure Duo Authentication Proxy for MFA

Configuring Duo authentication proxy for MFA involves these key steps (For more details, see https://duo.com/docs/authproxy-reference)

1. Download and install the latest Duo authentication proxy on your Windows or Linux machine.

   In this example, we have installed the primary authentication proxy on a Windows 2016 machine and the secondary proxy on an Ubuntu server.

2. Configure the proxy for your primary authenticator.

   Edit the Duo authentication proxy configuration file, authproxy.cfg, located in the conf subdirectory of the proxy installation path in the server using a text editor. You can add multiple ISE servers as RADIUS clients and multiple router subnets/IP addresses as part of the router.

3. Start the proxy server(s) and check the proxy logs for any configuration or connectivity error.

Note For installation on Windows, ensure sure that the Windows firewall is configured to allow connections for the authentication proxy.

Configure ISE for MFA

Configuring ISE for MFA involves these key steps (for more details, see Configure Duo Two Factor Authentication for ISE Management Access)

1. Integrate ISE with Duo authentication proxy:

   a. Add a new RADIUS token server:

      Administration > **Identity Management** > **External Identity Sources** > **RADIUS Token**, and click **Add**

      Ensure that the **Shared Secret** matches the one that you already defined in the *Configure Duo Authentication Proxy* task.

      For details, see step1 listed under ISE Configuration.

   b. Set the authentication method for the identity source:

      Navigate to **Administration** > **System** > **Admin Access** > **Admin Access** > **Authentication Method**, and select previously configured RADIUS token server (for example, **RADIUS:DUO**) as the **Identity Source**.

      For details, see Step 2 listed under ISE Configuration.

2. Create device admin policies:

a. Create a policy set:

   Navigate to **Work Centers** > **Device Administration** > **Device Admin Policy Sets**.

   In this example, we created a policy set that matches on both protocols (RADIUS and TACACS+) with the **Allowed Protocols** set to **Default Device Admin**.

b. Set the following policies inside the policy set:

   - **Authentication Policy**: In this example, we have set a default rule to check the Identity Source Sequence that we defined in the steps above which contains the RADIUS Token Servers (Duo Authentication Proxies) and Active Directory.
   - **Authorization Policy**: In this example, we have set a rule that checks if the authenticated user belongs either to the **Domain Users** or **NS-ISE-IOS-Admins** groups that we have configured in active directory (AD). If the user belongs to one of these groups, then the system returns the pre-configured **Command Sets** and **Shell Profile**.

3. Add and onboard users in Duo:

   You can configure Duo to automatically sync with your AD or manually add the user in Duo (for details, see Enroll user with Duo).

Configure RADIUS Server Attributes for MFA

This topic describes how to configure RADIUS server attributes for MFA on the Cisco IOS XR router (for more details, see configure-your-radius-client(s)).

Set the IP address of the RADIUS server to the IP address of your authentication proxy, the RADIUS server port to 1812, and the RADIUS secret to the appropriate secret that you configured in the *radius_server_auto* section in the authproxy.cfg file.

```
Router#configure
Router(config)#radius-server host 209.165.200.225auth-port 1812 acct-port 1813
Router(config-radius-host)#key test@1234
Router(config-radius-host)#commit
```

Verify MFA Set-up for SSH Connection

Once you complete the Duo MFA configurations, follow these steps to verify the set-up:

- Initiate an SSH connection from the SSH client router that is already added in the ISE, using the **ssh** command.
- Use the AD credentials for the admin user to log in.
- Upon successful authentication, confirm that the user received a **Duo Push/Passcode** notification on the Duo authentication device based on what is set in the Duo authentication proxy configuration file, authproxy.cfg.
- After approving the **Duo Push** or entering the correct Passcode, the admin user must be authenticated and authorized to access the router through the SSH connection.
- The live logs of RADIUS in the ISE server must show authentication requests against the Duo authentication proxies.

- Check the authproxy log file in your authentication proxy for any errors or issues.

SSH server timeouts

Unused connection timeout for SSH sessions

An unused connection timeout is a secure shell (SSH) configuration setting that

- terminates SSH connections with no active channels after a specified period
- prevents accumulation of stale connections, and
- protects routers from reaching SSH session limits due to inactive sessions.

An SSH connection is a secure communication session that

- establishes an encrypted link between a client and a server
- enables authentication and data exchange, and
- serves as the foundation for creating multiple SSH channels.

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
Unused connection timeout for SSH sessions	Release 25.3.1	You can prevent session limit exhaustion and maintain optimal system performance by automatically disconnecting SSH connections with no active channels. The feature introduces a configurable timeout for unused SSH connections, ensuring stale sessions do not occupy resources on your routers. The router monitors each SSH connection and terminates it when all channels remain closed and SSH clients do not create new channels within the configured timeout period. The feature introduces these changes: CLI: • **ssh server timeout** YANG Data Models: • `Cisco-IOS-XR-crypto-ssh-cfg.yang` data model was modified. • `Cisco-IOS-XR-um-ssh-cfg.yang` data model was modified. (see GitHub)

SSH connections form the basis for secure management and file transfer sessions in Cisco IOS XR routers. These connections persist after successful authentication and host one or more SSH channels for various operations. When all channels within a connection close and no new channels open, the unused connection timeout begins counting down.

The unused connection timeout functionality addresses issues where client automation scripts fail to properly close SSH connections, causing stale connections that remain active indefinitely and consume session resources.

Set unused connection timeout for SSH sessions

Follow this procedure to enable automatic disconnection of unused SSH connections on your routers to enhance device security and resource management.

Procedure

Step 1 Configure connection timeout for the unused SSH sessions on your router.

Example:

```
Router#configure
Router(config)#ssh server timeout connection 600
Router(config)#commit
```

The timeout begins counting down when all channels within a connection close and no new channels open.

Step 2 Verify the unused connection timeout configuration on the router.

Example:

```
Router#show run ssh
Thu Jul  3 09:45:41.201 UTC
ssh server v2
ssh server timeout
 connection 600
!
ssh server netconf vrf default
Router#


Router#show ssh server
Mon Aug 11 13:37:15.363 IST
---------------------
SSH Server Parameters
---------------------

Current supported versions := v2
                  SSH port := 22
                  SSH vrfs := vrfname:=default(v4-acl:=, v6-acl:=)
              Netconf Port := 830
              Netconf Vrfs := vrfname:=default(v4-acl:=, v6-acl:=)
        Netconf on SSH-port:= True (Netconf requests allowed on SSH port)

 Algorithms
---------------
        Hostkey Algorithms :=
x509v3-ssh-rsa,ssh-rsa-cert-v01@openssh.com,ssh-ed25519-cert-v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,


ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-ed25519,rsa-sha2-512,rsa-sha2-256,ssh-rsa,ssh-dss

   Key-Exchange Algorithms :=
ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group14-sha1,curve25519-sha256,


diffie-hellman-group14-sha256,diffie-hellman-group16-sha512,curve25519-sha256@libssh.org
     Encryption Algorithms :=
aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com

            Mac Algorithms := hmac-sha2-512,hmac-sha2-256

 Authentication Method Supported
------------------------------------
                PublicKey := Yes
                 Password := Yes
      Keyboard-Interactive := Yes
         Certificate Based := Yes

 Others
------------
                     DSCP  := 48
```

```
              Ratelimit  := 6000
           Sessionlimit  := 110
              Rekeytime  := 60
     Server rekeyvolume  := 1024
 TCP window scale factor  := 1
          Backup Server  := Disabled
        Host Trustpoint  :=
        User Trustpoint  :=
        Port Forwarding  := Disabled
Max Authentication Limit  := 20
    Certificate username  := Common name(CN)
 OpenSSH Host Trustpoint  :=
 OpenSSH User Trustpoint  :=
    OpenSSH HIBA enabled  := No
       OpenSSH HIBA role  :=
            HIBA version  := 1
Punt Rx SSH pkts to NetIO := False
         Channel Timeout  := 300
      Connection Timeout  := 600
```

Step 3 Check the SSH session details on the router to confirm that the unused SSH session disconnects after the specified period of inactivity.

Example:

This sample command output shows that there is an active channel and the connection timeout has not yet triggered.

```
Router#show ssh
Mon Aug 11 08:10:39.133 UTC
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host          ver
authentication connection type
-----------------------------------------------------------------------------------------------------------
Incoming sessions
2         0   vty0    0/RP0/CPU0      SESSION_OPEN    cisco     198.51.100.1  v2  password
       Command-Line-Interface

Outgoing sessions
```

The connection timeout triggers when there is no active channel. The **connection type** field then displays blank.

```
Router#show ssh
Mon Aug 11 08:10:47.675 UTC
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host          ver
authentication connection type
-----------------------------------------------------------------------------------------------------------
Incoming sessions
2         0   XXXXX   0/RP0/CPU0      SESSION_OPEN    cisco     198.51.100.1  v2  password


Outgoing sessions
Router#
```

This is the sample output when the connection timeout expires and entire connection closes.

```
Router#show ssh
Mon Aug 11 08:10:53.097 UTC
```

```
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host                  ver
authentication connection type
-------------------------------------------------------------------------------------------
Incoming sessions

Outgoing sessions
Router
```

Step 4 Check the system logs on the router console that indicates the termination of the inactive SSH sessions.

Example:

```
RP/0/RP0/CPU0:Aug 11 08:10:51.557 UTC: ssh_syslog_proxy[1215]:
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[42522]: terminating inactive connection
from user user1 198.51.100.1 port 25372
```

The router automatically disconnects unused SSH connections after the configured connection timeout period.

Channel timeout for SSH sessions

A channel timeout is a secure shell (SSH) configuration setting that

- closes an SSH channel if it remains idle for a specific duration
- applies to all types of channels, including Shell, SFTP, and Netconf, and
- ensures unused channels are closed promptly.

An SSH channel is a virtual communication path within an SSH connection that

- allows concurrent operations such as shell access or file transfer
- operates independently within its parent SSH connection, and
- can be opened and closed without affecting the overall connection.

SSH channels allow multiple management or file operations to run concurrently over a single SSH connection on the router.

Table 34: Feature History Table

Feature Name	Release Information	Feature Description
Channel timeout for SSH sessions	Release 25.3.1	You can improve resource efficiency and minimize potential security risks by automatically closing idle SSH channels on the routers after a specific period of inactivity. The feature introduces a configurable timeout for SSH channels which ensures that unused channels do not persist while the parent SSH connection remains active. The router monitors each SSH channel and closes any channel where no data is sent or received within the configured timeout period. The feature introduces these changes: CLI: • **ssh server timeout** YANG Data Models: • `Cisco-IOS-XR-crypto-ssh-cfg.yang` data model was modified. • `Cisco-IOS-XR-um-ssh-cfg.yang` data model was modified. (see GitHub)

The SSH client opens one or more SSH channels such as Shell, SFTP, and so on, within the SSH connection as needed. The channel timeout begins counting down after a specific period of inactivity.

Channel timeouts help enforce resource efficiency by closing idle channels, while the parent SSH connection remains active.

Set channel timeout for SSH sessions

Follow this procedure to enable automatic disconnection of idle SSH channels belonging to active SSH sessions on your routers to enhance device security and resource management.

Procedure

Step 1 Configure channel timeout for the SSH channels on your router.

Example:

```
Router#configure
Router(config)#ssh server timeout channel 300
Router(config)#commit
```

The timer expires when the SSH channel remains idle for the specified timeout period.

Step 2 Verify the SSH channel timeout configuration on the router by running these commands.

Example:

```
Router#show run ssh
Thu Jul  3 09:45:41.201 UTC
ssh server v2
ssh server timeout
 channel 300
 connection 600
!
ssh server netconf vrf default
Router#


Router#show ssh server
```

See the sample command output here.

Step 3 Check the SSH session details on the router to confirm if the idle SSH channel closes after the specified period of inactivity.

Example:

Before timer expiry:

```
Router#show ssh
Thu Jul  3 09:46:29.807 UTC
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host             ver
authentication connection type
-----------------------------------------------------------------------------------------------
Incoming sessions
8         0   vty0    0/RP0/CPU0      SESSION_OPEN    user1     198.51.100.1     v2  password
       Command-Line-Interface

Outgoing sessions
Router#
```

After timer expiry:

```
Router#show ssh
Thu Jul  3 09:51:33.818 UTC
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host             ver
authentication connection type
-----------------------------------------------------------------------------------------------
Incoming sessions
8         0   XXXXX   0/RP0/CPU0      SESSION_OPEN    user1     198.51.100.1     v2  password

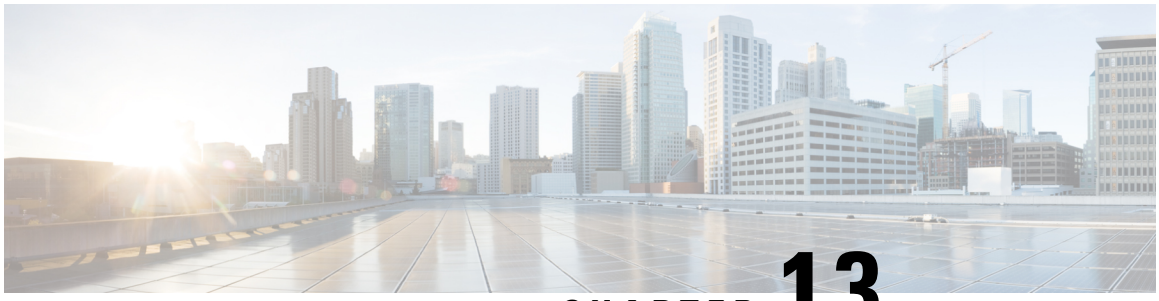

Outgoing sessions
Router#
```

Step 4 Check the system logs on the router console that indicates the closure of the idle SSH channels.

Example:

```
RP/0/RP0/CPU0:ios#RP/0/RP0/CPU0:Jul  3 09:46:32.002 UTC: ssh_syslog_proxy[1222]:
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[39648]: Closing channel 0 of user user1
198.51.100.1 port 34454 after 300 seconds of inactivity
```

The router automatically closes inactive SSH channels after the configured channel timeout period.



CHAPTER 13

Gated shell access

A gated shell access is a security feature that

- controls direct, interactive root shell access for supported shell commands,
- requires authorization using a Cisco or customer Consent Token (CT) path before restricting access, and
- stores in HWTAM Secure Object storage.

Consent Token authorization applies to the shell-access restriction operation. This feature uses a separate Consent Token challenge-response workflow for each shell session.

Table 35: Feature History Table

Feature Name	Release Information	Feature Description
Gated shell access	Release 26.3.1	Shell access restrictions require Consent Token authorization for direct root shell access initiated through supported shell commands.

- Key features of gated shell access, on page 255
- How gated shell access works, on page 256
- Restrict shell access using a Cisco-signed Consent Token, on page 257
- Unrestrict shell access using a Cisco-signed Consent Token, on page 258
- Restrict shell access using an owner Consent Token, on page 259
- Unrestrict shell access using an owner Consent Token, on page 260
- Terminate a pending Consent Token authorization request, on page 261
- Access a restricted shell, on page 261

Key features of gated shell access

Use this reference to determine whether a platform supports gated shell access and to identify the associated restrictions and CLI commands.

Key features:

- Gates only direct shell access via supported CLI commands (such as **bash**, **run**, and **attach**). Indirect access through scripts or internal processes is not gated in the initial release.

- It enables restriction or unrestriction of gated shell access, ensuring that only authorized personnel can change shell-access policies.
- Depends on the consent token infrastructure to be present and supported on the platform for operation.
- The restricted-shell selection is stored in an HWTAM secure object.

Table 36: Gated shell access CLI commands

Command	Purpose
platform security shell-access restrict challenge {cisco | owner}	Initiates the Consent Token challenge for the selected Cisco or customer authorization path.
platform security shell-access restrict response <response_string>	Submits the Consent Token response to authorize and complete the shell-access restriction operation.
platform security shell-access restrict terminate-auth	Terminates the pending Consent Token authorization request for shell-access restriction.
platform security shell-access unrestrict challenge {cisco | owner}	Starts the Consent Token authorization flow to remove the restriction.
platform security shell-access unrestrict response <response_string>	Submits the Consent Token response for unrestriction.
platform security shell-access unrestrict terminate-auth	Terminates the pending unrestriction authorization request operation.
show platform security shell-access status	Displays the restricted-shell configuration and handshake status.

The HWTAM Secure Object holds the restricted-shell selection and handshake state. The restricted-shell selection is preserved across reboots and reloads.

How gated shell access works

The process occurs when a user restricts direct interactive shell access on a platform that supports the Consent Token infrastructure.

Summary

When a user restricts shell access, the system records the restriction state in the HWTAM Secure Object. Subsequent supported direct, interactive root shell access attempts check this state and are denied while the restriction is active.

Workflow

These stages describe the gated shell access workflow:

1. A user restricts shell access or requests access to a direct, interactive root shell.
2. The system stores the restricted-shell access state in the HWTAM Secure Object and applies the restriction to supported direct, interactive root shell access.

3. The user must complete the consent token response challenge handshake.
4. The user enters the Consent Token response. When the response succeeds, the system permits one root shell instance for that user session.
5. A user starts the unrestriction authorization flow, submits the response, and the system returns to the default non-gated root shell behavior after successful authorization.

Restrict shell access using a Cisco-signed Consent Token

Use this task to restrict supported direct, interactive root shell access by using a Cisco-signed Consent Token.

Procedure

Step 1 Generate the challenge string on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access restrict challenge cisco
Thu Aug 20 06:06:33.888 UTC

+--------------------------------------+
Node location: node0_RP0_CPU0
+--------------------------------------+
Challenge string:
<challenge_string>
```

This produces a challenge string containing the device ID, a nonce, and the requested action.

Step 2 Submit the challenge string to a Cisco TAC engineer. Cisco verifies that the requester is the legitimate device owner and is authorized to perform the requested action. If verified, the TAC engineer provides a signed response string.

Step 3 Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access restrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Restrict Shell Access in node0_RP0_CPU0
```

Example:

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, this restricts the direct shell access.

Note

- To access the root shell after restricting shell access, see Access a restricted shell, on page 261.

If required, to terminate a pending Consent Token authorization request for shell-access restriction, see Terminate a pending Consent Token authorization request, on page 261.

Unrestrict shell access using a Cisco-signed Consent Token

Use this task to unrestrict the supported direct, interactive root shell access by using a Cisco-signed Consent Token.

Before you begin

Ensure that gated shell access is restricted. For more information about restricting shell access by using a Cisco-signed Consent Token, see Restrict shell access using a Cisco-signed Consent Token, on page 257.

Procedure

Step 1 Generate an unrestrict challenge on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict challenge
Thu Aug 20 06:06:33.888 UTC

+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
```

Step 2 Submit the challenge string to a Cisco TAC engineer. Cisco verifies that the requester is the legitimate device owner and is authorized to perform the requested action. If verified, the TAC engineer provides a signed response string.

Step 3 Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Unrestrict Shell Access in node0_RP0_CPU0
```

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, the response removes the restriction on direct shell access.

Note
If required, to terminate a pending Consent Token handshake for shell-access restriction, see Terminate a pending Consent Token authorization request, on page 261.

Step 4 Verify the shell-access status.

Example:

```
RP/0/RP0/CPU0:ios# show platform security shell-access status
Thu Aug 20 05:20:01.690 UTC
Restricted shell access enabled: No
```

Restrict shell access using an owner Consent Token

Use this task to restrict supported direct, interactive root shell access by using an owner Consent Token.

Procedure

Step 1 Generate the challenge string on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access restrict challenge owner
Thu Aug 20 06:06:33.888 UTC

+--------------------------------------+
Node location: node0_RP0_CPU0
+--------------------------------------+
Challenge string:
<challenge_string>
```

This produces a challenge string containing the device ID, a nonce, and the requested action.

Step 2 To generate the response string owner key type , refer to Provisioning customer consent token guide.

Step 3 Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access restrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Restrict Shell Access in node0_RP0_CPU0
```

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, this restricts the direct shell access.

Note

- To access the root shell after restricting shell access, see Access a restricted shell, on page 261.

  If required, to terminate a pending Consent Token authorization request for shell-access restriction, see Terminate a pending Consent Token authorization request, on page 261.

Unrestrict shell access using an owner Consent Token

Restore unrestricted root shell access by removing the restricted-shell setting, allowing access to advanced system features when necessary.

Use this task to unrestrict the supported direct, interactive root shell access by using an owner Consent Token.

Before you begin

Ensure that gated shell access is restricted. For more information about restricting shell access by using an owner Consent Token, see Restrict shell access using an owner Consent Token, on page 259.

Procedure

Step 1 Generate an unrestrict challenge on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict challenge
Thu Aug 20 06:06:33.888 UTC
+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
```

Step 2 To generate the response string owner key type, refer to Provisioning customer consent token guide.

Step 3 Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example:

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Unrestrict Shell Access in node0_RP0_CPU0
```

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, the response removes the restriction on direct shell access.

Note
If required, to terminate a pending Consent Token handshake for shell-access restriction, see Terminate a pending Consent Token authorization request, on page 261.

Step 4 Verify the shell-access status.

Example:

```
RP/0/RP0/CPU0:ios# show platform security shell-access status
Thu Aug 20 05:20:01.690 UTC
Restricted shell access enabled: No
```

Terminate a pending Consent Token authorization request

Use this task to terminate a pending Consent Token authorization request for restrict or unrestrict shell access.

Use this task only after generating a Consent Token challenge and before submitting a response. The terminate command applies to both Cisco and owner Consent Token requests.

Procedure

Step 1 Generate a Consent Token challenge for either restrict or unrestrict shell access, as described in the applicable task.

Do not submit the response.

Step 2 To terminate a pending Consent Token handshake:

- For restrict shell access, use `platform security shell-access restrict terminate-auth` command.

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios# platform security shell-access restrict challenge cisco
Thu Aug 20 06:08:45.228 UTC

+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
RP/0/RP0/CPU0:ios# platform security shell-access restrict terminate-auth
Thu Aug 20 06:09:10.334 UTC
Successfully terminated challenge session for Restrict Shell Access in node0_RP0_CPU0
```

- For unrestrict shell access, use `platform security shell-access unrestrict terminate-auth` command.

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict challenge cisco
Thu Aug 20 06:08:45.228 UTC

+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict terminate-auth
Thu Aug 20 06:09:10.334 UTC
Successfully terminated challenge session for Unrestrict Shell Access in node0_RP0_CPU0
```

The system confirms that it terminated the pending authorization request.

Access a restricted shell

Use this task to access a restricted, direct, interactive root shell by completing the Consent Token challenge-response workflow.

This task applies to supported direct, interactive root shell commands, such as `bash`, `run` or `attach`.

Procedure

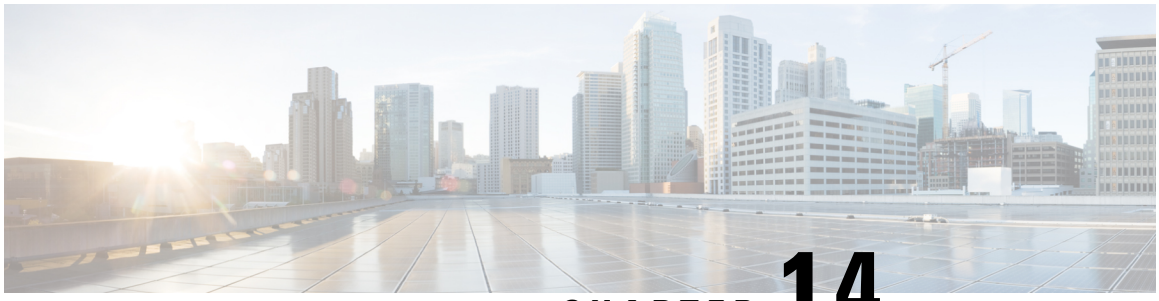
Step 1 Restrict shell access by using either a Cisco-signed Consent Token or an Owner Consent Token.

Step 2 Enter the `bash`, `run` or `attach` command. When prompted, complete the `cisco` or `owner` Consent Token challenge-response.

A valid response permits the requested shell instance.

Example:

```
RP/0/RP0/CPU0:ios# bash
Thu Aug 20 06:01:15.407 UTC
Restricted shell access is enabled. Please obtain a consent token to proceed.
Consent token challenge: <challenge_string>
Enter challenge response, or Ctrl-C to terminate:
<response_string>
Challenge response accepted successfully.
```



CHAPTER 14

Implementing Lawful Intercept

Lawful intercept is the lawfully authorized interception and monitoring of communications of an intercept subject. Service providers worldwide are legally required to assist law enforcement agencies in conducting electronic surveillance in both circuit-switched and packet-mode networks.

Only authorized service provider personnel are permitted to process and configure lawfully authorized intercept orders. Network administrators and technicians are prohibited from obtaining knowledge of lawfully authorized intercept orders, or intercepts in progress. Error messages or program messages for intercepts installed in the router are not displayed on the console.

Lawful Intercept is not a part of the Cisco IOS XR software by default. You have to install it separately by installing and activating **ncs560-li-1.0.0.0-r66136I.x86_64.rpm**.

For more information about activating and deactivating the Lawful Intercept package, see the Installing Lawful Intercept (LI) Package, on page 266 section.

- Interception Mode, on page 263
- Data Interception, on page 264
- Lawful Intercept Topology, on page 264
- Benefits of Lawful Intercept, on page 265
- Information About Lawful Intercept Implementation, on page 265
- Prerequisites for Implementing Lawful Intercept, on page 265
- Installing Lawful Intercept (LI) Package, on page 266
- Types of Lawful Intercept Mediation Device, on page 267
- Restrictions for Implementing Lawful Intercept, on page 267
- Limitations of Lawful Intercept, on page 269
- How to Configure SNMPv3 Access for Lawful Intercept, on page 269
- Additional Information on Lawful Intercept, on page 271

Interception Mode

The lawful intercept operates in the **Global LI** mode.

In this mode, the taps are installed on all the line cards in the ingress direction. The lawful intercept is available on line cards where QoS peering is enabled. With the global tap, the traffic for the target can be intercepted regardless of ingress point. Only the tap that has wild cards in the interface field is supported.

Data Interception

Data are intercepted in this manner:

- The MD initiates communication content intercept requests to the content IAP router using SNMPv3.
- The content IAP router intercepts the communication content, replicates it, and sends it to the MD in IPv4 UDP format.
- Intercepted data sessions are sent from the MD to the collection function of the law enforcement agency, using a supported delivery standard for lawful intercept.

Information About the MD

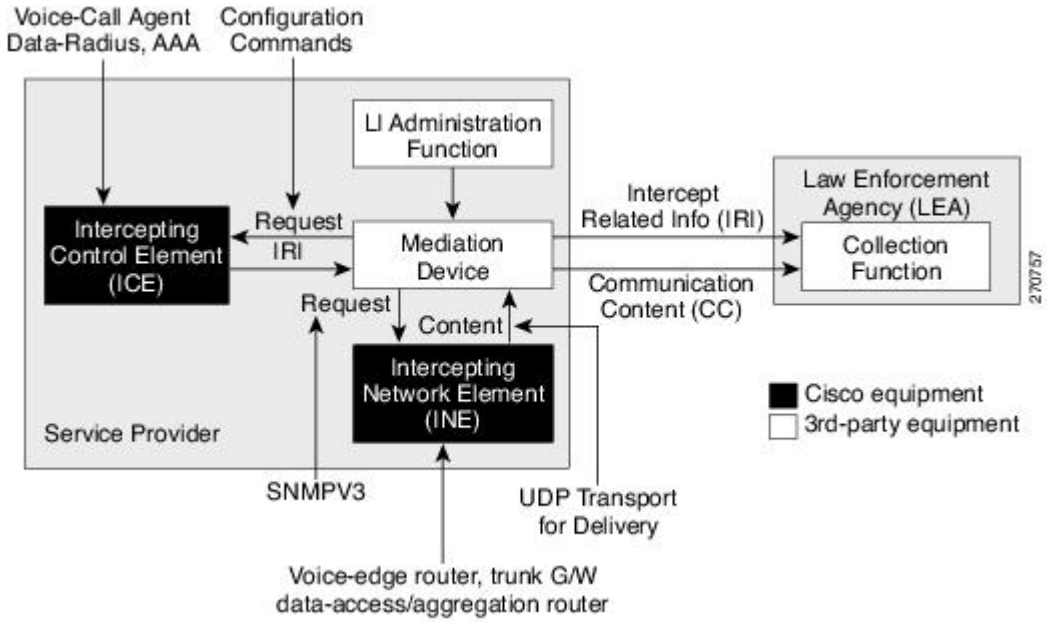
The MD performs these tasks:

- Activates the intercept at the authorized time and removes it when the authorized time period elapses.
- Periodically audits the elements in the network to ensure that:
  - *only* authorized intercepts are in place.
  - *all* authorized intercepts are in place.

Lawful Intercept Topology

This figure shows intercept access points and interfaces in a lawful intercept topology for both voice and data interception.

Figure 8: Lawful Intercept Topology for Both Voice and Data Interception



Note

- The router will be used as content Intercept Access Point (IAP) router, or the Intercepting Network Element (INE) in lawful interception operation.
- The Intercepting Control Element (ICE) could be either a Cisco equipment or a third party equipment.

Benefits of Lawful Intercept

Lawful intercept has the following benefits:

- Allows multiple LEAs to run a lawful intercept on the same Router without each other's knowledge.
- Does not affect subscriber services on the router.
- Supports wiretaps in both the input and output direction.
- Supports wiretaps of Layer 3 traffic.
- Cannot be detected by the target.
- Uses Simple Network Management Protocol Version 3 (SNMPv3) and security features such as the View-based Access Control Model (SNMP-VACM-MIB) and User-based Security Model (SNMP-USM-MIB) to restrict access to lawful intercept information and components.
- Hides information about lawful intercepts from all but the most privileged users. An administrator must set up access rights to enable privileged users to access lawful intercept information.

Information About Lawful Intercept Implementation

Cisco lawful intercept is based on RFC3924 architecture and SNMPv3 provisioning architecture. SNMPv3 addresses the requirements to authenticate data origin and ensure that the connection from the router to the Mediation Device (MD) is secure. This ensures that unauthorized parties cannot forge an intercept target.

Lawful intercept offers these capabilities:

- SNMPv3 lawful intercept provisioning interface
- Lawful intercept MIB: CISCO-TAP2-MIB, version 2
- CISCO-IP-TAP-MIB manages the Cisco intercept feature for IP and is used along with CISCO-TAP2-MIB to intercept IP traffic
- IPv4 user datagram protocol (UDP) encapsulation to the MD
- Replication and forwarding of intercepted packets to the MD

Prerequisites for Implementing Lawful Intercept

Lawful intercept implementation requires that these prerequisites are met:

- The router is used as content Intercept Access Point (IAP) router in lawful interception operation.
- **Provisioned Router—**The router must be already provisioned.

Tip For the purpose of lawful intercept taps, provisioning a loopback interface has advantages over other interface types.

- **Management Plane Configured to Enable SNMPv3—**Allows the management plane to accept SNMP commands, so that the commands go to the interface (preferably, a loopback interface) on the router. This allows the mediation device (MD) to communicate with a physical interface.
- **VACM Views Enabled for SNMP Server—**View-based access control model (VACM) views must be enabled on the router.
- **Provisioned MD—**For detailed information, see the vendor documentation associated with your MD.
- The MD uses the **CISCO-TAP2-MIB** to set up communications between the router acting as the content IAP, and the MD. The MD uses the **CISCO-IP-TAP-MIB** to set up the filter for the IP addresses and port numbers to be intercepted.
- The MD can be located anywhere in the network but must be reachable from the content IAP router, which is being used to intercept the target. MD should be reachable *only* from global routing table and *not* from VRF routing table.

Installing Lawful Intercept (LI) Package

As LI is not a part of the Cisco IOS XR image by default, you need to install it separately.

Installing and Activating the LI Package

Use the **show install committed** command in EXEC mode to verify the committed software packages.

To install the Lawful Intercept (LI) package, you must install and activate the **ncs560-li-1.0.0.0-r66136I.x86_64.rpm** rpms.

Procedure

Step 1 **Configuration**

```
Router# install add source
tftp://223.255.254.252/auto/tftp-sjc-users/username/ncs560-li-1.0.0.0-r66136I.x86_64.rpm
Router# install activate ncs560-li-1.0.0.0-r66136I.x86_64.rpm
Router# install commit
```

Step 2 **Verification**

```
Router# show install active
Node 0/RP0/CPU0 [RP]
```

```
  Boot Partition: xr_lv0
 Active Packages: 2
       ncs560-xr-6.6.1.36I version=6.6.1.36I [Boot image]
       ncs560-li-1.0.0.0-r66136I.x86_64.rpm

Node 0/0/CPU0 [LC]
  Boot Partition: xr_lcp_lv0
  Active Packages: 2
       ncs560-xr-6.6.1.36I version=6.6.1.36I [Boot image]
       ncs560-li-1.0.0.0-r66136I.x86_64.rpm
```

Deactivating the LI RPM

Note If you deactivate or uninstall the Lawful Intercept package, it may cause flapping of some control protocols which result in traffic drop. So, ensure to plan accordingly.

To uninstall the Lawful Intercept package, deactivate **ncs560-li-1.0.0.0-r66136I.x86_64.rpm** as shown in the following steps:

Configuration

```
Router# install deactivate ncs560-li-1.0.0.0-r66136I.x86_64.rpm
Router# install commit
Router# install remove ncs560-li-1.0.0.0-r66136I.x86_64.rpm
Router# show install committed
```

Types of Lawful Intercept Mediation Device

There are two types of Lawful Intercept mediation device.

- **MD reachable via IPv4:** The destination (mediation device) for intercepted packets are accessed through an IPv4 path.
- **MD reachable via MPLS:** The destination (mediation device) for intercepted packets are accessed through MPLS (Multiprotocol Label Switching) path.

Note **MD reachable via MPLS** is supported only on routers that have Cisco NC57 line cards.

Restrictions for Implementing Lawful Intercept

The following restrictions are applicable for Lawful Intercept:

- Lawful Intercept shares a pool of 16 unique source IP addresses with tunnel-ip. The combined configuration of GRE tunnel-ips and the MDs (the cTap2MediationSrcInterface field) shall not yield more than 16 unique source IPs. Note that when configuring the MD, if the value 0 is passed in for the cTap2MediationSrcInterface field, it will be resolved into a source IP address, which is the egress IP to the MD destination.
- Lawful intercept is supported only to match layer 3 IPv4/IPv6 packets.
- One Tap-to-multiple MDs is not supported.
- Only ingress packet tapping is supported.
- After the route processor reload or fail-over, the MD and Tap configuration must be re-provisioned.
- Only IPv4 MD is supported.
- MD should be reachable via default vrf.
- The path to the MD must have the ARP resolved. Any other traffic or protocol will trigger ARP.
- MD next-hop must have ARP resolved. Any other traffic or protocol will trigger ARP.
- Lawful Intercept has no intersection with the GRE Tunnel feature, except that they allocate hardware resources (16 unique egress IP addresses) from the same pool. In the normal case, the egress interface for the LI packets is decided by the forwarding algorithm. No resource is needed from that unique address pool. However, if the Lawful Intercept configuration mandates that the Lawful Intercept packets have to egress through a certain interface (the cTap2MediationSrcInterface field in the MD configuration), then the forwarding module must be configured so that the packets go out through that interface. In that case, a resource must be allocated from the unique address pool. If GRE uses up all resources, then LI does not work.
- Lawful Intercept Stats is not supported.
- Even though the original packets can be fragmented, the LI packets cannot be fragmented. The MTU of the egress interface to the MD must be large enough to support the size of the packets captured.
- Lawful intercept does not provide support for these features on the router:
    - IPv4/IPv6 multicast tapping
    - IPv6 MD encapsulation
    - Per interface tapping
    - Tagged packet tapping
    - Replicating a single tap to multiple MDs
    - Tapping L2 flows and SRv6 traffic
    - RTP encapsulation
    - Lawful Intercept and SPAN on the same interface

Limitations of Lawful Intercept

The following are some limitations of Lawful Intercept.

- Only 250 MDs and 500 Taps of IPv4 and IPv6 each are supported.
- In the case of fragmented packets, only the first fragment will be intercepted.
- LI traffic cannot be transmitted over the management interface.

Scale or Performance Values

The router support the following scalability and performance values for lawful intercept:

- A maximum of 500 IPv4 intercepts and 500 IPv6 intercepts are supported.
- The scale decreases, if port ranges are used in the taps.
- The IPv6 entries consume double the memory of the IPv4 entries. Hence, the IPv6 scale will reduce to half of the IPv4 scale.
- Interception rate is 1 Gbps best effort per Linecard NPU.

How to Configure SNMPv3 Access for Lawful Intercept

Perform these procedures to configure SNMPv3 for the purpose of Lawful Intercept enablement:

Disabling SNMP-based Lawful Intercept

Lawful Intercept is enabled by default on the router after installing and activating the **ncs560-li-1.0.0.0-r66136I.x86_64.rpm** rpms.

- To disable Lawful Intercept, enter the **lawful-intercept disable** command in global configuration mode.
- To re-enable it, use the **no** form of this command.

Disabling SNMP-based Lawful Intercept: Example

```
Router# configure
Router(config)# lawful-intercept disable
```



Note The **lawful-intercept disable** command is available on the router, only after installing and activating the **ncs560-li-1.0.0.0-r66136I.x86_64.rpm** rpms.

All SNMP-based taps are dropped when lawful intercept is disabled.

Configuring the Inband Management Plane Protection Feature

If MPP was not earlier configured to work with another protocol, then ensure that the MPP feature is also not configured to enable the SNMP server to communicate with the mediation device for lawful interception. In such cases, MPP must be configured specifically as an inband interface to allow SNMP commands to be accepted by the router, using a specified interface or all interfaces.



Note Ensure this task is performed, even if you have recently migrated to Cisco IOS XR Software from Cisco IOS, and you had MPP configured for a given protocol.

For lawful intercept, a loopback interface is often the choice for SNMP messages. If you choose this interface type, you must include it in your inband management configuration.

Example: Configuring the Inband Management Plane Protection Feature

This example illustrates how to enable the MPP feature, which is disabled by default, for the purpose of lawful intercept.

You must specifically enable management activities, either globally or on a per-inband-port basis, using this procedure. To globally enable inbound MPP, use the keyword **all** with the **interface** command, rather than use a particular interface type and instance ID with it.

```
router# configure
router(config)# control-plane
router(config-ctrl)# management-plane
router(config-mpp)# inband
router(config-mpp-inband)# interface loopback0
router(config-mpp-inband-Loopback0)# allow snmp
router(config-mpp-inband-Loopback0)# commit
router(config-mpp-inband-Loopback0)# exit
router(config-mpp-inband)# exit
router(config-mpp)# exit
router(config-ctr)# exit
router(config)# exit
router# show mgmt-plane inband interface loopback0
Management Plane Protection - inband interface
interface - Loopback0
        snmp configured -
All peers allowed
router(config)# commit
```

Enabling the Lawful Intercept SNMP Server Configuration

The following SNMP server configuration tasks enable the Cisco LI feature on a router running Cisco IOS XR Software by allowing the MD to intercept data sessions.

Configuration

```
router(config)# snmp-server engineID local 00:00:00:09:00:00:00:a1:61:6c:20:56
router(config)# snmp-server host 1.75.55.1 traps version 3 priv user-name udp-port 4444
router(config)# snmp-server user user-name li-group v3 auth md5 clear lab priv des56 clear
 lab
router(config)# snmp-server view li-view ciscoTap2MIB included
router(config)# snmp-server view li-view ciscoIpTapMIB included
```

```
router(config)# snmp-server view li-view snmp included
router(config)# snmp-server view li-view ifMIB included
router(config)# snmp-server view li-view 1.3.6.1.6.3.1.1.4.1 included
router(config)# snmp-server group li-group v3 auth read li-view write li-view notify li-view
```



Note SNMP configuration must be removed while deactivating the LI RPM.

Additional Information on Lawful Intercept

Intercepting IPv4 and IPv6 Packets

This section provides details for intercepting IPv4 and IPv6 packets supported on the router.

Lawful Intercept Filters

The following filters are supported for classifying a tap:

- IP address type
- Destination address
- Destination mask
- Source address
- Source mask
- ToS (Type of Service) and ToS mask
- L4 Protocol
- Destination port with range
- Source port with range
- VRF (VPN Routing and Forwarding)



Note Flow-id and inteface filters are not supported.

Encapsulation Type Supported for Intercepted Packets

Intercepted packets mapping the tap are replicated, encapsulated, and then sent to the MD. IPv4 and IPv6 packets are encapsulated using IPv4 UDP encapsulation. The replicated packets are forwarded to MD using UDP as the content delivery protocol.

The intercepted packet gets a new UDP header and IPv4 header. Information for IPv4 header is derived from MD configuration. Apart from the IP and UDP headers, a 4-byte channel identifier (CCCID) is also inserted

after the UDP header in the packet. The router does not support forwarding the same replicated packets to multiple MDs.



Note Encapsulation types, such as RTP and RTP-NOR, are not supported.

High Availability for Lawful Intercept

High availability for lawful intercept provides operational continuity of the TAP flows and provisioned MD tables to reduce loss of information due to route processor fail over (RPFO).

To achieve continuous interception of a stream, when RP fail over is detected; MDs are required to re-provision all the rows relating to CISCO-TAP2-MIB and CISCO-IP-TAP-MIB to synchronize database view across RP and MD.

Preserving TAP and MD Tables during RP Fail Over

At any point in time, MD has the responsibility to detect the loss of the taps via SNMP configuration process.

After RPFO is completed, MD should re-provision all the entries in the stream tables, MD tables, and IP taps with the same values they had before fail over. As long as an entry is re-provisioned in time, existing taps will continue to flow without any loss.

The following restrictions are listed for re-provisioning MD and tap tables with respect to behavior of SNMP operation on citapStreamEntry, cTap2StreamEntry, cTap2MediationEntry MIB objects:

- After RPFO, table rows that are not re-provisioned, shall return NO_SUCH_INSTANCE value as result of SNMP Get operation.
- Entire row in the table must be created in a single configuration step, with exactly same values as before RPFO, and with the rowStatus as CreateAndGo. Only exception is the cTap2MediationTimeout object, that should reflect valid future time.

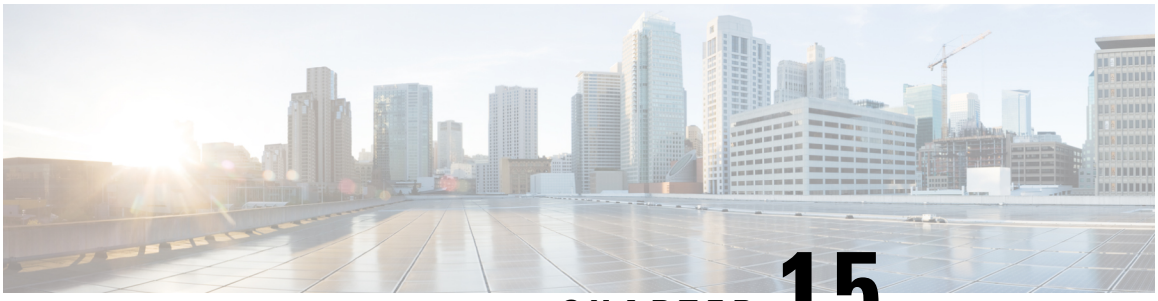
Replay Timer

The replay timer is an internal timeout that provides enough time for MD to re-provision tap entries while maintaining existing tap flows. It resets and starts on the active RP when RPFO takes place. The replay timer is a factor of number of LI entries in router with a minimum value of 10 minutes.

After replay timeout, interception stops on taps that are not re-provisioned.



Note In case high availability is not required, MD waits for entries to age out after fail over. MD cannot change an entry before replay timer expiry. It can either reinstall taps as is, and then modify; or wait for it to age out.



CHAPTER 15

Cisco MASA Service

Table 37: Feature History Table

Feature Name	Release Information	Feature Description
Cisco MASA Service	IOS XR 7.8.1	The Cisco Manufacturer Authorized Signing Authority (MASA) service creates ownership vouchers (OVs) for a Cisco IOS XR router. These OVs along with the owner certificate (OC) **certify** that the router belongs to a given customer. Use cases where OVs and OCs are required include secure ZTP workflows and securely booting up your device on a 5G cell site over a third-party ethernet service. You can use the MASA service to download, and view logging and audit of OVs for the routers you own. This service also enables Cisco's Account teams to assign the serial number of a device to customers and view details of the logging, verification, and audit of OVs.

Key Terms and Concepts

Authentication Flow: The purpose of the Authentication flow is to identify and authenticate the router when it boots up. During this flow, the router also checks if the network can be trusted. The router does this by:

- validating the OV it received during the bootstrapping process and
- verifying the signature on the onboarding information with the owner certificate it received during the bootstrapping process.

The workflow involves the router booting to dynamically obtain OV from Manufacturer Authorized signing Authority (MASA).

MASA Service: There are many services that require the ownership of the router to be authenticated, so it can be trusted by the network. MASA is a service run by Cisco to create and log OVs that are then used to validate the ownership of the router.

Owner Certificate: The OC is an X.509 certificate [RFC5280] that is used to identify an *owner*, for example, an organization. The OC can be signed by any certificate authority (CA).

The OC is used by a router to verify the CA signature using the public key that is also in the owner certificate.

The OC structure must contain the owner certificate itself, as well as all intermediate certificates leading to the "pinned-domain-cert" (PDC) certificate specified in the ownership voucher.

Ownership Voucher: The ownership voucher (OV) [RFC8366] is used to securely identify the router's owner, as known to the manufacturer. The ownership voucher is signed by the device's manufacturer.

The OV is used to verify that the owner certificate has a chain of trust leading to the trusted certificate (PDC) included in the ownership voucher.

pinned-domain-cert: The PDC field present in the OV typically pins a domain certificate, such as the certificate of a domain CA.

- Why Do I Need Cisco MASA?, on page 274
- Use Cases for Ownership Vouchers, on page 274
- Authentication Flow, on page 275
- Interacting with the MASA Server, on page 277
- Workflow to Provision a Router Using Ownership Voucher, on page 284

Why Do I Need Cisco MASA?

The Cisco MASA service securely authorizes ownership of a router so that the router can then establish a secure connection to the router owner's (your) network infrastructure.

The establishment of the ownership of the router is achieved through an authentication workflow that on successful completion generates an ownership voucher (OV). The primary purpose of the OV is to securely convey a certificate—the "pinned-domain-cert" (PDC), that the router can then use to authenticate subsequent interactions with the network, for example, secure bootstrapping. Establishing ownership is important to the bootstrapping mechanisms so that the router can authenticate the network that is trying to take control of it.

Use Cases for Ownership Vouchers

- **Secure Zero Touch Provisioning (ZTP) Bootstrapping**

  Secure ZTP requires the ability to securely bootstrap a router over an untrusted network. This requires the ability of MASA to provide an OV to the router. The OV is used to authenticate the router to ensure connectivity of the router to the network.

  For more information on Secure ZTP, see the Secure Zero Touch Provisioning chapter in the *System Setup and Software Installation Guide for NCS 5500 Series Routers*.

Note MASA can help generate OVs for Cisco Routers only.

- **Application Hosting on XR**

  Cisco IOS XR's Application Hosting (App Hosting) capability provides an IOS XR container on the router. This allows an application that augments XR features to be deployed. These applications can fall in one of the following categories:

  - Customer Apps—developed by Cisco's customers and cannot be signed by Cisco.
  - Partner Apps—developed by partners and are signed by Cisco.
  - Cisco App—developed by Cisco and signed by Cisco.

  You can use MASA in conjunction with the Golden ISO Tool (gisobuild.py) to provide the OVs to enable secure workflows for onboarding third party RPMs on router running Cisco IOS XR.

  For more information, see the *Application Hosting Guide for Cisco NCS 5500 Series Routers*.

- **Deploy Router Using BootZ**

  Bootz is a secure zero-touch provisioning solution for data centers that automates the setup of network devices while ensuring robust security. It enables devices to connect and authenticate with the Bootz server, safeguarding the onboarding process against unauthorized access and cyber threats, streamlining remote device configuration without compromising safety.

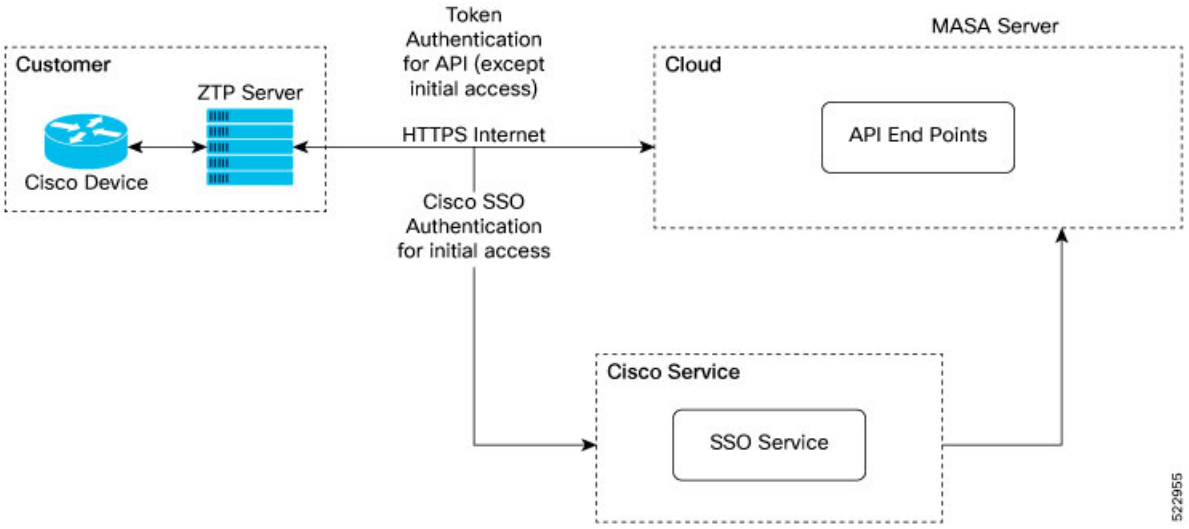
  Bootz uses a MASA to issue OVs that authenticate network devices during zero-touch provisioning.

  For more information on BootZ, see the *System Setup and Software Installation Guide for NCS 5500 Series Routers*.

Authentication Flow

The following figure is a high-level overview of different components involved in the authentication flow.

Figure 9: Components of the Authentication Flow



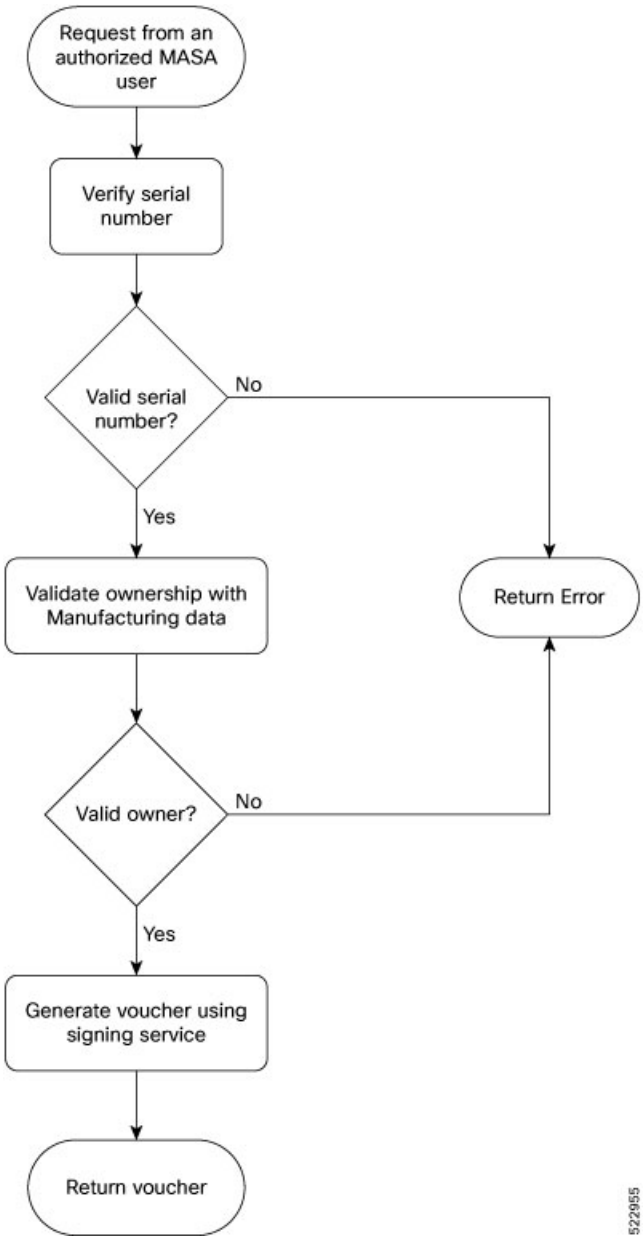
You can interact with the MASA Server web application through the ZTP Server to request, manage, and download the OVs for your routers.

The Zero Touch Provisioning (ZTP) server is to used to make a REST API call to the MASA Server.

The MASA Server authenticates the user, and on successful validation, generates the OVs.

The following figure illustrates the typical workflow to obtain the OVs.

Figure 10: Workflow to Obtain Ownership Vouchers



Interacting with the MASA Server

There are two ways to interact with the MASA server:

- through Web Application
- through REST API calls

Entities

The following entities interact with the MASA Server:

- **Organization**—A group in MASA specific to a Cisco customer. Data and access for each Organization is available to members of that group only.
- **Admin**—One or more initially-designated member(s) of an Organization who can invite other members into that organization in MASA, set access restrictions, and adjust other organization level settings.
- **User**—Any non-admin member of an organization who can interact with MASA. A user must be invited into an organization by the Admin
  - By default, new users have view-only access.
  - The Admin assigns permissions to request, download, or archive ownership vouchers

Prerequisites for Interacting with MASA Server

1. You must be an authorized MASA User
   - You must have a Cisco account and an active invitation to access MASA for the first time.

Note Contact the Cisco Technical Assistance team or your Account team to get a Cisco account.

   - Initial authentication requires *Cisco Single Sign On* to the MASA web application (masa.cisco.com).

     For subsequent authentication, you can generate access keys called *tokens*. Token serve as an alternative authentication mechanism that can be passed along in the header of API calls.

Note To generate access keys for the first time, on masa.cisco.com, go to **Settings**→ **Tokens**. For subsequent sessions, use API calls to manage existing tokens or create new ones as long as an unexpired token is still available.

     The following is an example of using a token in a header of a REST API call.

```
‘Authorization: Bearer
637c98ddcc58c75f679a94d7f244777be05c6600923c4549bc5669b26e04f2bc
gAAAAABjfRr9hqndFqbuqes9OvcfgucApgxpmm9qoVmUidYEs-_Aziu7yue-10dazZ3Rxk6vJHYD2Je7Z-IOD1Zc7kYSuBTX0
6GcQvF2e3nSM-_F9BoltjxAHcXkoMgbqS4APFGi16LiWRyP2b1_0rZO-EaTKFLEldTLfMAmovPDkZZ5vbBwRS058PZN1vB3IZIZ
jftYYYi9H_grazfwnAImjKbQC6tjQw==
```

     Tokens can have a custom validity period of up to six months that can be revoked at any time. The scope of the tokens is limited to scope of your role.

2. ZTP server must be able to access the Internet

Note MASA application is served through HTTPS to provide a secure connection between the end user and the service.

User Permissions

The MASA Server supports Role Based Access Control and provides the following access:

- Regular user—By default, regular users have only read access to their organization. Admin users can provide additional privileges as required.
- Admin—Admin users have the ability to view and manage OVs for all routers in the database in their organization as well as other privileges as mentioned in the table below.

Table 38: User Permissions

Type	Regular User	Admin
Invite other People into the organization	Not allowed	Allowed by default
Add or remove permissions for other users	Not allowed	Allowed by default
View all existing vouchers	Allowed by default	Allowed by default
Request new vouchers	Permission can be provided by Admin	Allowed by default
Download vouchers	Permission can be provided by Admin	Allowed by default
Archive vouchers	Permission can be provided by Admin	Allowed by default

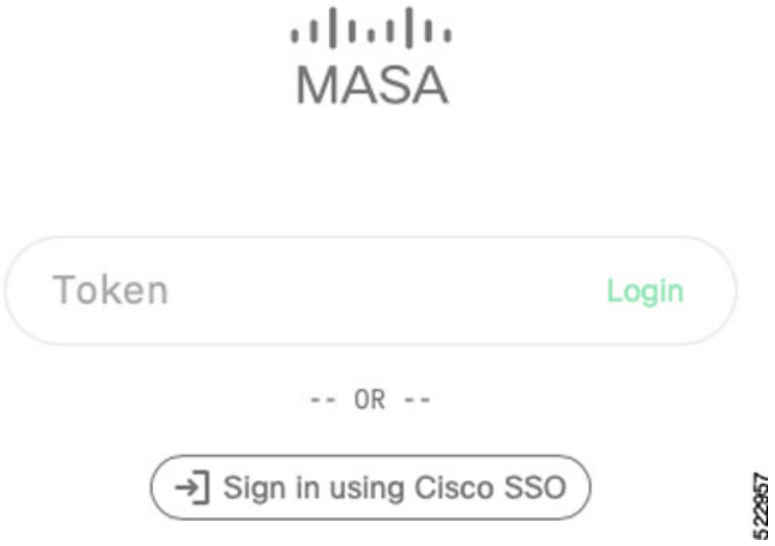
Interacting with MASA Through Web Application

Table 39: Feature History Table

Feature Name	Release Information	Feature Description
Pre-upload Pinned-Domain Certificate	Release 24.1.1	You can now pre-upload your Pinned-Domain Certificate (PDC) credentials before requesting OVs Ownership Vouchers (OVs) from the MASA server, thus making the voucher request process easier.

1. Go to masa.cisco.com

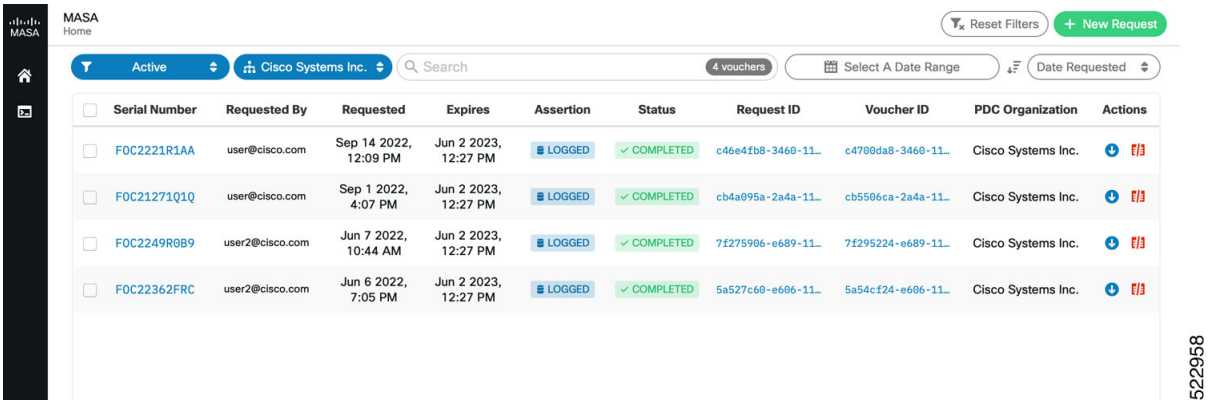
Figure 11: Sign in Page—MASA Web Application



2. Click **Sign in using Cisco** SSO.
3. Enter your username and password to access the application
4. Accept the End User License Agreement.

   The MASA Home page displays the status of any recent requests that were initiated and quick links to download any recently generated ownership vouchers.

Figure 12: Home Page—MASA Web Application



Requesting OVs for Your Router

1. Click **New Request** on the top right of the Home page.
2. In the New Request dialog box, enter details for one of the following:
   - Serial number of your router

     You can get the serial number from the bottom of your router; it is an 11 digit alphanumeric string. You can also get the serial number by running the **show inventory** command on your router.

- Pinned-domain Certificate

  There are multiple ways to generate a PDC (.pem). For example, through OpenSSL. You can either paste the content of the certificate directly or browse to a file that contains the PDC.

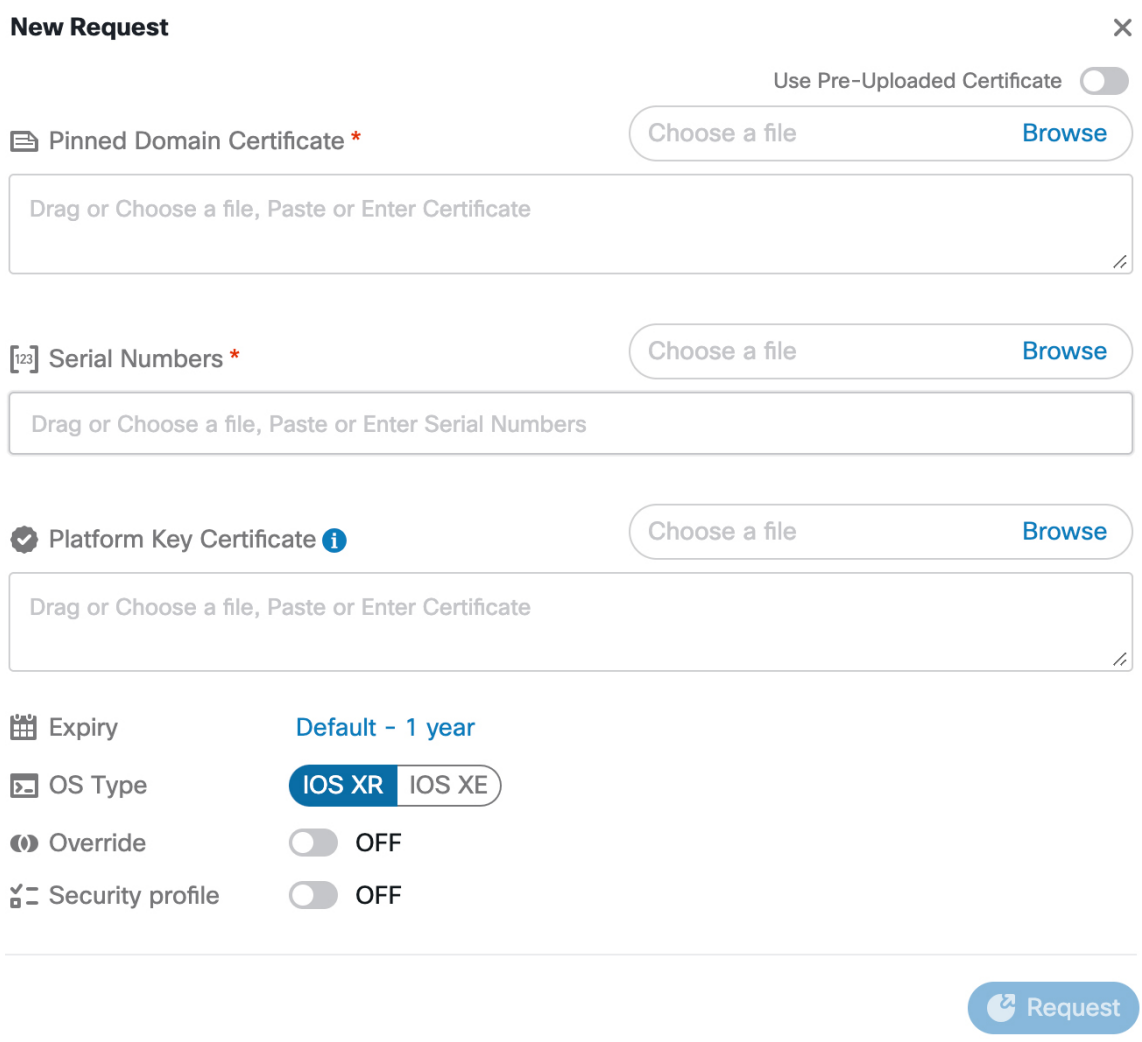
  You can pre-upload the certificate prior to requesting the OV.

  To select the pre-uploaded certificate while requesting OV, turn on the toggle button named *use pre-uploaded certificate*.You can see the already uploaded certificates here, you can select the certificate from this list.

- Serial number of one or more routers for which you want the OVs.

Note Always use the serial number of the chassis of your router.

Figure 13: New Request Page



523805

Figure 14: Home Page—With New OVs Displayed



522960

Depending on your user permissions, you can perform the following actions from the Home page.

- Download the generated OVs.
- Regenerate OVs.
- View details of past requests
- Filter, sort, and group the requests based on their attributes
- Archive the OVs.

Interacting with MASA Through REST APIs

You can also use APIs to programmatically interact with the MASA service.

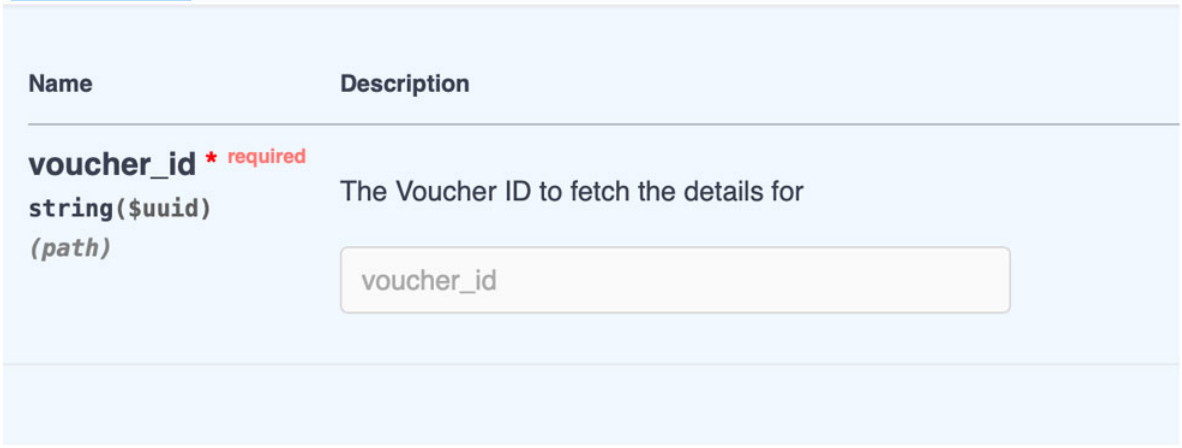
See the OpenAPI documentation page that contains details about the paths, formats, and structures of the APIs.

For example, use this API to request for the ownership voucher:

```
POST /request/ov
```

Use this API to fetch details about an already generated voucher:

```
GET /voucher/{voucher_id}
```




```
Response:
{
  "ok": true,
  "voucher": {
    "req_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "voucher_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "requested_at": "2022-08-31T09:43:39.719Z",
    "created_at": "2022-08-31T09:43:39.719Z",
    "expires_at": "2022-08-31T09:43:39.719Z",
    "last_renewal_at": "2022-08-31T09:43:39.719Z",
    "assertion": "logged",
    "status": "completed",
    "serial_number": "T8I52J1IKOM",
    "pdc_organization": "Cisco Systems",
    "requested_by": "user1@cisco.com"
  }
}
```

Note "serial Number" is serial number of the route processor. You can provide up to 20 serial numbers in a single request.

Interaction with MASA through gRPC

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
Interaction with MASA through gRPC	Release 24.1.1	From this release, you can use the gRPC protocol to interact with MASA APIs in addition to the current HTTP protocols. Through structured serialization of data with gRPC's Protocol Buffers, the communication between services is made more efficient, type-safe, and consistent.

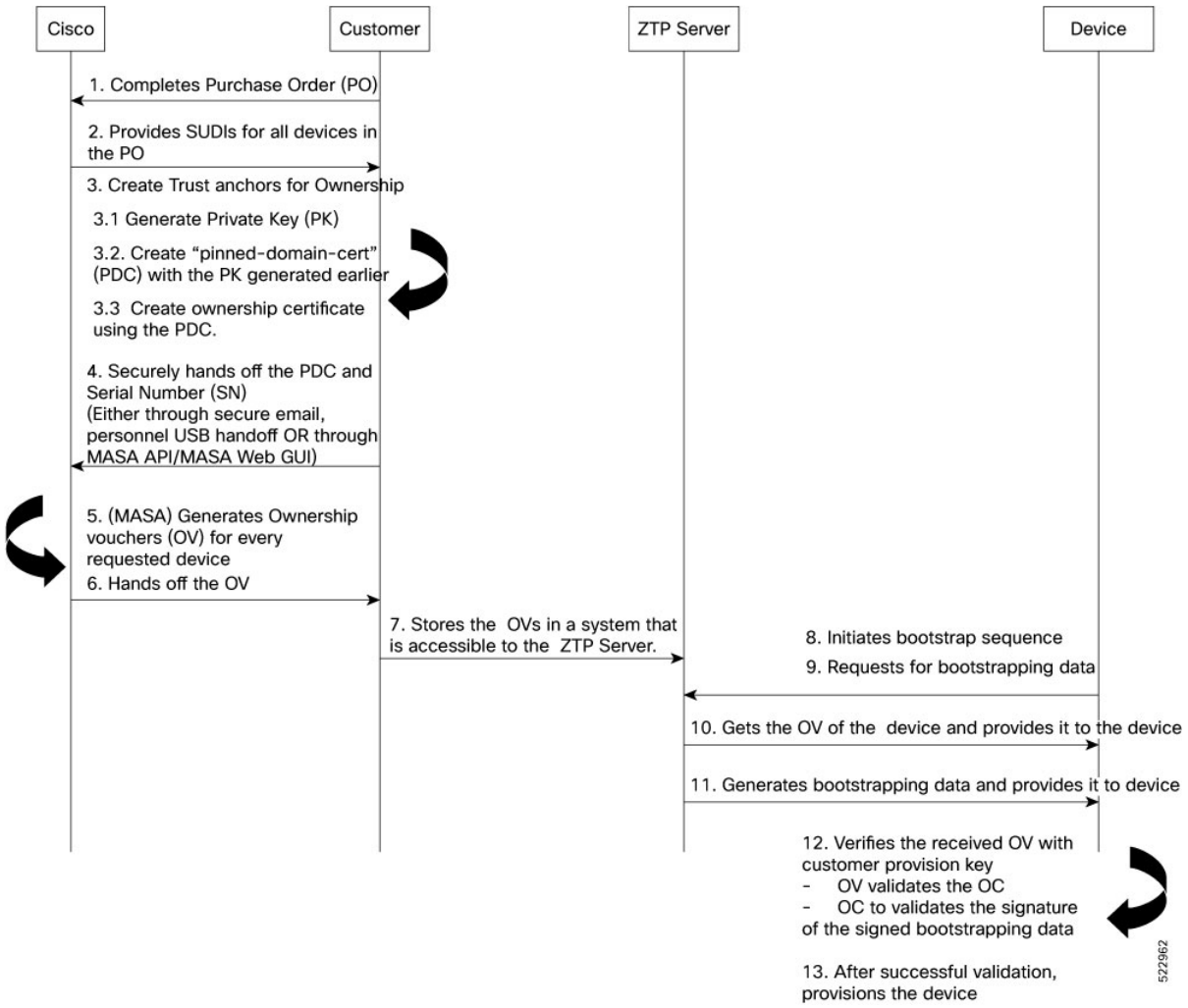
The following MASA APIs are accessible using gRPC protocol in addition to http protocol:

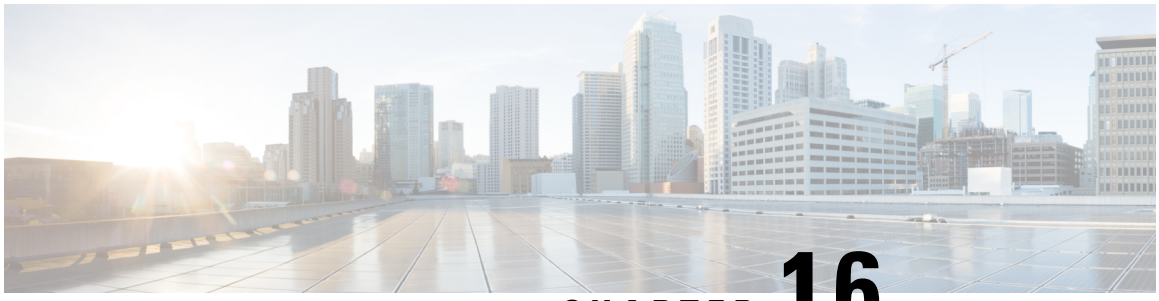
RPC	Description
rpc GetGroup	Returns the domain-certificates (keyed by id), serials, and user/role mappings for that group.
rpc AddUserRole	Assigns a role to a user in a named group. Username is unique to an Org ID.
rpc RemoveUserRole	Removes a role from a user in a named group. Username is unique to an Org ID.
rpc GetUserRole	Returns the roles that the user is assigned in the group. Username is unique to an Org ID. A user can only view roles of another user in the group that it has a role assigned to.
rpc CreateDomainCert	Creates the certificate in the group.
rpc GetDomainCert	Reveals the details of the certificate.
rpc DeleteDomainCert	Deletes the certificate from the database.
rpc GetOwnershipVoucher	Issues an ownership voucher.

For more information on gRPC, see Use gRPC Protocol to Define Network Operations with Data Models in the *Programmability Configuration Guide for NCS 5500 Series Routers.*

Workflow to Provision a Router Using Ownership Voucher

The following figure illustrates the complete workflow to provision a Cisco IOS XR router by using the ownership vouchers.


Cisco
Customer
ZTP Server
Device
1. Completes Purchase Order (PO)
2. Provides SUDIs for all devices in the PO
3. Create Trust anchors for Ownership
3.1 Generate Private Key (PK)
3.2. Create "pinned-domain-cert" (PDC) with the PK generated earlier
3.3 Create ownership certificate using the PDC.
4. Securely hands off the PDC and Serial Number (SN) (Either through secure email, personnel USB handoff OR through MASA API/MASA Web GUI)
5. (MASA) Generates Ownership vouchers (OV) for every requested device
6. Hands off the OV
7. Stores the OVs in a system that is accessible to the ZTP Server.
8. Initiates bootstrap sequence
9. Requests for bootstrapping data
10. Gets the OV of the device and provides it to the device
11. Generates bootstrapping data and provides it to device
12. Verifies the received OV with customer provision key
- OV validates the OC
- OC to validates the signature of the signed bootstrapping data
13. After successful validation, provisions the device
522962



CHAPTER 16

EST protocol for Automated Certificate Provisioning

This feature allows you to enable the EST (Enrollment Over Secure Transport) protocol for all trustpoints while using TLS to secure transport.

- EST protocol for automated certificate provisioning, on page 287
- Configure the EST protocol, on page 292

EST protocol for automated certificate provisioning

Enrollment over Secure Transport (EST) is a digital certificate provisioning protocol that

- enhances security by using TLS for secure communication, and
- automates the renewal of certificates to minimize manual intervention.

Table 41: Feature History Table

Feature Name	Release Information	Feature Description
EST protocol for automated certificate provisioning	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This release introduces support for Enrollment over Secure Transport (EST), a digital certificate provisioning protocol, which enhances certificate management by offering secure transport using TLS and designated certificate requestors. It enables automated certificate renewal. EST is an enhancement of the existing Simple Certificate Enrollment Protocol (SCEP), providing improved security and flexibility for certificate management operations over both IPv4 and IPv6 networks. The feature introduces these changes: **CLI**: • The **method-est** keyword is introduced in the **crypto ca trustpoint** command. • The **client-authentication** command is introduced. • The **SSL-profile** command is introduced. **YANG Data Model**: • `Cisco-IOS-XR-crypto-cepki-cfg` • `Cisco-IOS-XR-um-crypto-cfg` (see GitHub, YANG Data Models Navigator)

EST protocol—key features and benefits

These are the key features and benefits of the EST protocol:

- **Secure transport**—EST utilizes TLS to ensure that all messages and certificates are securely transmitted without the need for additional encapsulation.
- **Designated certificate requestors**—With EST, the certificate signing request (CSR) can be associated with a specific trusted requestor that is authenticated with TLS.
- **Automated certificate renewal**—The protocol supports automatic re-enrollment, facilitating seamless renewal of certificates.

EST protocol—key components and related terminologies

Understand key components and terms related to the EST protocol:

- **Certificate Management Protocol (CMP)**—A protocol with comprehensive management capabilities for managing digital certificates, including their enrollment, renewal, and revocation.
- **Certificate signing request (CSR)**—A message sent by the client to the CA to request a digital certificate. It includes the client's public key and identifying information.
- **PKCS 7**—A standard used for cryptographic message syntax, which is employed by SCEP to encapsulate messages.
- **Public Key Infrastructure (PKI)**—The framework that manages digital certificates and keys, essential for certificate provisioning operations using the EST protocol.
- **Simple Certificate Enrollment Protocol (SCEP)**—An older protocol used for certificate provisioning that relies on HTTP and PKCS 7 for securing messages and lacks some of the modern security features of EST.

Client authentication options in the EST protocol

- TLS certificate-based authentication (mTLS)—Requires the client to have a valid certificate chain signed by a CA in the EST server trust store so that server can verify the client during mutual TLS (mTLS) authentication. The bootstrap certificate is used only during the first enrollment and can be onboarded using existing methods like SCEP. For re-enrollment, the certificate issued during the initial enrollment is used in the mTLS handshake.
- HTTP-based authentication—When a bootstrap certificate is unavailable for initial enrollment, the EST client establishes a TLS session and then sends HTTP authentication credentials. This HTTP-based authentication method uses Type6 encryption to protect passwords configured on the router, preventing unauthorized access. By default, Type6 encryption is disabled and must be enabled manually. Additionally, a master key must be created and securely stored in the Trust Anchor Module (TAM). During this process, the client transmits a username and clear text password through the TLS-encrypted channel, ensuring the password remains secure due to the TLS protection. To validate the server during this TLS session, the device must have the server certificate's CA/subordinate CA chain installed locally. This validation confirms the authenticity of the EST server and secures the communication channel before the HTTP authentication details are sent..

Guidelines and limitations for configuring the EST protocol

Supported key types

- In IOS-XR release 25.1.1, only RSA keys are supported for signing the Certificate Signing Request (CSR) during enrollment; ECDSA keys are not.

Client authentication methods

- The EST client supports both TLS certificate-based authentication and HTTP-based authentication. Cisco recommends using TLS certificate-based authentication as per RFC 7030, even though both methods are available.

TLS validation and authentication requirements

- If client or server certificate validation fails during the TLS handshake, EST enrollment fails. No fallback mechanism switches to HTTP authentication if TLS certificate-based client authentication fails.

Re-enrollment

- For re-enrollment, the client always uses the previously issued certificate to establish the TLS connection.
- A re-enrollment profile is required in scenarios where the EST server uses a fixed username and password for re-enrollment, but uses a one-time password (OTP) for initial enrollment. This behavior is optional and depends on how the EST server is configured.

EST Client Support Requires Multiple Trustpoints

The EST client support requires multiple trustpoints because it enables secure certificate enrollment over TLS for all trustpoints configured on the device, and a single trustpoint configuration is insufficient to support this functionality.

Certificate types and their uses in EST protocol

Learn about the types of certificates used in the EST protocol, along with their issuers and specific uses.

Table 42: Certificates and their corresponding uses

If the certificate type is...	And the issuer is...	Then it is used for...
EST server certificate	EST server	authenticating the CA during the TLS handshake when the EST server interacts with clients.
EST client certificate	EST client	authenticating to the EST server during certificate enrollment operations.
End entity or leaf certificate	EST server	non-EST uses, such as authenticating devices in different contexts beyond EST operations.
EST server certificate	third-party web CA	providing authentication for the EST server by a CA that is not part of the EST infrastructure but trusted as a third party.

If the certificate type is...	And the issuer is...	Then it is used for...
third-party EST client certificate	third-party web CA	authenticating the EST client to the EST server for initial interactions before enrollment.

Trust anchor databases in EST protocol

Effective certificate validation and authentication in the EST protocol involve different types of trust anchor (TA) databases, each serving specific purposes.

A TA database refers to a repository of trusted anchor certificates used to verify the authenticity of a Certification Authority (CA) during the certificate enrollment process. The database acts as a trusted source for validating the CA issuing certificates to clients through the EST protocol.

Table 43: Trust anchor databases and their uses

If the TA database type is...	Then it is used by...	For the purpose of...
EST server	EST server	authenticating certificates issued by the EST CA during enroll and re-enroll operations.
EST server (implicit)	EST server	authenticating certificates issued by third-party TAs; can be disabled if not needed.
EST client	EST client	authenticating EST server certificates issued by the EST CA.
EST client (implicit)	EST client	authenticating an EST server using an externally issued certificate; can be disabled if unnecessary.

Pre-defined message types in EST protocol

The EST protocol includes several pre-defined message types, each serving distinct functions to facilitate secure certificate provisioning:

- **CACerts**—This message type is used to request the CA certificates. It ensures the client can validate the CA's authenticity before proceeding with certificate enrollment.
- **SimpleEnroll**—Utilized for simple enrollment of a certificate, this message type involves a client sending a certificate signing request (CSR) to the CA. If the request is valid, the CA issues a certificate in response.
- **Reenroll**—Designed for requesting certificate renewal or reissuance, this message type allows clients to obtain a new certificate with updated validity or information when an existing certificate is about to expire.

Configure the EST protocol

Use this procedure to configure the EST protocol on your router, enabling secure and automated certificate provisioning.

Before you begin

Before configuring the EST protocol, ensure

- access to the EST server is established and operational,
- required certificates and keys are available for configuration,
- you meet specific encryption requirements if you must use HTTP-based authentication:
  - You must create a rsa key using the **crypto key generate rsa key** *key-name* command. Use the generated key to configure the EST_TRUSTPOINT.
  - Type6 encryption is enabled using the command **password6 encryption aes** for secure password handling, and
  - You must create a master key and store it in Trust Anchor Module (TAM) using the command **key config-key password-encryption** to facilitate encryption.

Note A master key for Type 6 encryption is typically stored securely within the device internal memory, specifically within a protected area known as the TAM.

Procedure

Step 1 Select the authentication method.

You can configure either TLS certificate-based authentication, HTTP-based authentication, or both. Use the table below to determine the appropriate sub-step based on the availability of a bootstrap certificate.

Table 44: Bootstrap certificate availability and authentication method

If...	Then follow...
A bootstrap certificate is not available	HTTP-Based Authentication sub-step.
A bootstrap certificate is available	TLS-Based Authentication sub-step.

a) For HTTP-based authentication, create client authentication profiles for enrollment and re-enrollment.

Example:

```
Router(config)#client-authentication profile P1_enroll
Router(config-client-authentication-profile)#username estuser password Encrypt6
estpwd_enrol
Router(config-client-authentication-profile)#exit
```

```
Router(config)#client-authentication profile P2_re-enroll
Router(config-client-authentication-profile)#username estuser password Encrypt6
estpwd_re_enrol
Router(config-client-authentication-profile)#commit
Router(config-client-authentication-profile)#exit
Router(config)#crypto ca trustpoint EST_TRUSTPOINT
Router(config-trustp)#method est
Router(config-trustpoint)#enrollment url https://192.168.30.1:port
Router(config-trustpoint)#authentication-profile P1_enroll
Router(config-trustpoint)#re-enrollment authentication-profile P2_re-enroll
Router(config-trustpoint)#rsakeypair est-client
Router(config-trustpoint)#commit
```

You can use ipv4, ipv6, and hostname addresses as enrollment URL.

Table 45: Re-enrollment profile scenarios for HTTP-based authentication

If...	Then...
separate profiles are configured for enrollment and re-enrollment	*P1_enroll* is used for enrollment and *P2_re-enroll* is used for re-enrollment.
only the enrollment profile is configured	*P1_enroll* is used for both enrollment and re-enrollment.

If you are using the multi-tier CA certificate system, the CA root certificate need to be imported in another trustpoint using terminal, SCEP or local file method.

b) For TLS-based authentication, configure an SSL profile with the bootstrap certificate.

Example:

```
Router# configure terminal
Router(config)# ssl profile EST_SSL_profile
Router(config-SSL-profile)# certificate EST_BOOTSTRAP_TP
Router(config-SSL-profile)# commit
```

For initial enrollment, ensure you add a bootstrap certificate using existing methods like terminal, SCEP, and so on.

Step 2 Configure a PKI trustpoint to specify the enrollment method as EST and attach the authentication profiles.

The code example includes CLI commands to attach both HTTP and TLS-based authentication profiles.

This command initializes the EST method for the specified trustpoint.

Example:

```
Router#configure terminal
Router(config)#crypto ca trustpoint EST-TP
Router(config)#ssl-profile EST_enroll
Router(config-trustp)#method est
Router(config-method-est)#commit
Router(config-trustp)#enrollment authentication-profile P1_enroll
Router(config-trustp)#re-enrollment authentication-profile P2_re-enroll
Router(config-trustp)#commit
```

Step 3 Authenticate and enroll the certificates.

Run the CLI commands in EXEC mode.

```
Router# crypto ca authenticate EST_TP
Router# crypto ca enroll EST_TP
```

Step 4 Verify the certificate enrollment using EST is successful.

Example:

```
Router# show run crypto ca trustpoint EST_TP:
crypto ca trustpoint est_tp
 ssl-profile example
 method est
 !
 enrollment authentication-profile enroll
 re-enrollment authentication-profile re_enroll
!
```

Step 5 Verify the certificates are enrolled on the trustpoint.

Example:

```
Router# show crypto ca certificates EST_TP

Trustpoint       : EST_TP
================================================
CA certificate
  Serial Number  : 10:01
  Subject:
            CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
            CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 12:31:40 UTC Sun Jun 14 2020
  Validity End   : 12:31:40 UTC Wed Jun 12 2030

  CRL Distribution Point
            http://10.105.236.78/crl.der
  SHA1 Fingerprint:
             D8E0C11ECED96F67FDBC800DB6A126676A76BD62
Trusted Certificate Chain
  Serial Number  : 0F:A0:06:7A:C9:5E:A9:E7:61:A2:B9:2B:27:D1:D6:8F:3D:51:43:3B
  Subject:
            CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
            CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 13:12:32 UTC Sun Jun 07 2020
  Validity End   : 13:12:32 UTC Sat Jun 02 2040

  CRL Distribution Point
            http://10.105.236.78/crl.der
  SHA1 Fingerprint:
             08E71248FB7578614442E713AC87C461D173952F
Router certificate
  Key usage      : General Purpose
  Status         : Available
  Serial Number  : 28:E5
  Subject:
            CN=test
  Issued By      :
            CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 08:49:54 UTC Mon Feb 06 2023
  Validity End   : 08:49:54 UTC Wed Mar 08 2023
  SHA1 Fingerprint:
             6C8644FA67D9CEBC7C5665C35838265F578835AB
Associated Trustpoint: EST_TP
```