



System Security Configuration Guide for Cisco NCS 540 Series Routers, IOS XR Release 7.3.x

First Published: 2021-02-26

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 NON-INFRINGEMENT 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)

© 2021 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1

Configuring AAA Services 1

Configuring AAA Services	1
Prerequisites for Configuring AAA Services	1
Restrictions for Configuring AAA Services	2
Configure Task group	2
Configure User Groups	3
Configure First User on Cisco Routers	5
Configure Users	6
Password Masking For Type 7 Password Authentication	8
Configure Type 8 and Type 9 Passwords	8
Configure Type 10 Password	10
Backward Compatibility for Password Types	11
Configure AAA Password Policy	12
Configure Password Policy for User Secret and Password	14
Configure Router to RADIUS Server Communication	17
Configure RADIUS Dead-Server Detection	20
Configure TACACS+ Server	21
Configure RADIUS Server Groups	24
Configure TACACS+ Server Groups	26
Configure Per VRF TACACS+ Server Groups	27
Create Series of Authentication Methods	29
Create Series of Authorization Methods	31
Create Series of Accounting Methods	33
Generate Interim Accounting Records	35
Apply Method List	36
Enable Accounting Services	37

Configure Login Parameters 38
 Task Maps 39
 Overview on AAA Services 40
 Model-based AAA 57

CHAPTER 2

Implementing Certification Authority Interoperability 65

Implementing Certification Authority Interoperability 65
 Prerequisites for Implementing Certification Authority 65
 Restrictions for Implementing Certification Authority 66
 Configure Router Hostname and IP Domain Name 66
 Generate RSA Key Pair 67
 Import Public Key to the Router 68
 Declare Certification Authority and Configure Trusted Point 68
 Authenticate CA 70
 Request Your Own Certificates 71
 Configure Certificate Enrollment Using Cut-and-Paste 72
 Certificate Authority Trust Pool Management 75
 CA Certificate Bundling in the Trust Pool 75
 Prerequisites for CA Trust Pool Management 75
 Restrictions for CA trust pool management 75
 Updating the CA Trustpool 75
 Configuring Optional Trustpool Policy Parameters 77
 Handling of CA Certificates appearing both in Trust Pool and Trust Point 78
 Expiry Notification for PKI Certificate 78
 Learn About the PKI Alert Notification 78
 Enable PKI Traps 79
 Regenerate the Certificate 80
 Integrating Cisco IOS XR and Crosswork Trust Insights 81
 How to Integrate Cisco IOS XR and Crosswork Trust Insights 81
 Generate Key Pair 84
 Generate System Trust Point for the Leaf and Root Certificate 85
 Generate Root and Leaf Certificates 86
 System Certificates Expiry 88
 Collect Data Dossier 88

Procedure to Test Key Generation and Data-signing with Different Key Algorithm	91
Verify Authenticity of RPM Packages Using Fingerprint	92
Support for Ed25519 Public-Key Signature System	94
Generate Crypto Key for Ed25519 Signature Algorithm	95
Integrate Cisco IOS XR with Cisco Crosswork Trust Insights using Ed25519	95
Information About Implementing Certification Authority	96
Supported Standards for Certification Authority Interoperability	96
Certification Authorities	97

CHAPTER 3**Implementing Keychain Management 99**

Implementing Keychain Management	99
Restrictions for Implementing Keychain Management	99
Configure Keychain	99
Configure Tolerance Specification to Accept Keys	101
Configure Key Identifier for Keychain	102
Configure Text for Key String	102
Determine Valid Keys	103
Configure Keys to Generate Authentication Digest for Outbound Application Traffic	104
Configure Cryptographic Algorithm	106
Lifetime of Key	107

CHAPTER 4**Implementing Type 6 Password Encryption 109**

How to Implement Type 6 Password Encryption	109
Enabling Type6 Feature and Creating a Primary Key (Type 6 Server)	109
Implementing Key Chain for BGP Sessions (Type 6 Client)	112
Creating a BGP Session (Type 6 Password Encryption Use Case)	113

CHAPTER 5**Understanding MACSec Encryption 115**

MKA Authentication Process	115
MACsec Frame Format	116
Advantages of Using MACsec Encryption	117
Hardware Support for MACSec	117
MACsec PSK	118
Fallback PSK	118

Configuring and Verifying MACsec Encryption	119
Creating a MACsec Keychain	119
Creating a User-Defined MACsec Policy	122
Applying MACsec Configuration on an Interface	124
MACsec Policy Exceptions	126
How to Create MACsec Policy Exception	126
Verifying MACsec Encryption on IOS XR	127
Verifying MACsec Encryption on the Router	133

CHAPTER 6**802.1X Port-Based Authentication 137**

Restrictions for IEEE 802.1X Port-Based Authentication	137
IEEE 802.1X Device Roles	138
Understanding 802.1X Port-Based Authentication	138
802.1X host-modes	139
Configure 802.1X host-modes	140
Prerequisites for 802.1X Port-Based Authentication	140
802.1X with Remote RADIUS Authentication	140
Configure RADIUS Server	140
Configure 802.1X Authentication Method	141
Configure 802.1X Authenticator Profile	141
Configure 802.1X Profile on Interface	142
802.1X with Local EAP Authentication	142
Generate RSA Key Pair	142
Configure Trustpoint	143
Configure Domain Name	143
Certificate Configurations	143
Configure EAP Profile	144
Configure 802.1X Authenticator Profile	144
Configure 802.1X Profile on Interface	145
Router as 802.1X Supplicant	145
Configure 802.1X Supplicant Profile	146
Configure 802.1X Profile on Interface	146
Verify 802.1X Port-Based Authentication	146
Show Command Outputs	146

Syslog Messages 148

CHAPTER 7**MACSec Using EAP-TLS Authentication 149**

Configure MACSec Encryption Using EAP-TLS Authentication 150

Configure MACSec EAP on an Interface 150

Verify MACSec EAP Configuration on an Interface 151

CHAPTER 8**Understanding URPF 153**

Configuring URPF Loose Mode 153

CHAPTER 9**Implementing Management Plane Protection 155**

Implementing Management Plane Protection 155

Benefits of Management Plane Protection 156

Restrictions for Implementing Management Plane Protection 156

Configure Device for Management Plane Protection for Inband Interface 156

Configure Device for Management Plane Protection for Out-of-band Interface 159

Information About Implementing Management Plane Protection 163

Peer-Filtering on Interfaces 163

Control Plane Protection 163

Management Plane 163

CHAPTER 10**Traffic Protection for Third-Party Applications 165**

gRPC Protocol 165

Limitations for Traffic Protection for Third-Party Applications 166

Prerequisites for Traffic Protection for Third-Party Applications Over GRPC 166

Configuring Traffic Protection for Third-Party Applications 167

Troubleshooting Traffic Protection for Third-Party Applications 167

CHAPTER 11**Implementing Secure Shell 169**

Implementing Secure Shell 169

Information About Implementing Secure Shell 170

SSH Server 170

SSH Client 170

SFTP Feature Overview	171
RSA Based Host Authentication	172
RSA Based User Authentication	173
SSHv2 Client Keyboard-Interactive Authentication	174
Prerequisites for Implementing Secure Shell	174
SSH and SFTP in Baseline Cisco IOS XR Software Image	174
CiscoSSH	176
Guidelines for Using CiscoSSH	179
Restrictions for Implementing Secure Shell	182
Configure SSH	183
Automatic Generation of SSH Host-Key Pairs	187
Configure the Allowed SSH Host-Key Pair Algorithms	187
Ed25519 Public-Key Signature Algorithm Support for SSH	189
How to Generate Ed25519 Public Key for SSH	190
Configure SSH Client	190
Configuring CBC Mode Ciphers	192
SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm	193
Disable HMAC Algorithm	195
Enable Cipher Public Key	195
User Configurable Maximum Authentication Attempts for SSH	197
Configure Maximum Authentication Attempts for SSH	198
X.509v3 Certificate-based Authentication for SSH	199
Configure X.509v3 Certificate-based Authentication for SSH	202
Importing Public Key	206
SSH Port Forwarding	208
How to Enable SSH Port Forwarding	210

CHAPTER 12**Implementing Lawful Intercept 213**

Information About Lawful Intercept Implementation	214
Prerequisites for Implementing Lawful Intercept	214
Restrictions for Implementing Lawful Intercept	215
Lawful Intercept Topology	216
Benefits of Lawful Intercept	216
Installing Lawful Intercept (LI) Package	217

Installing and Activating the LI Package	217
Deactivating the LI RPM	218
How to Configure SNMPv3 Access for Lawful Intercept	218
Disabling SNMP-based Lawful Intercept	218
Configuring the Inband Management Plane Protection Feature	219
Enabling the Lawful Intercept SNMP Server Configuration	220
Additional Information on Lawful Intercept	220
Interception Mode	220
Data Interception	221
Scale or Performance Values	221
Intercepting IPv4 and IPv6 Packets	221
Lawful Intercept Filters	221
Encapsulation Type Supported for Intercepted Packets	222
High Availability for Lawful Intercept	222
Preserving TAP and MD Tables during RP Fail Over	222
Replay Timer	223

CHAPTER 13

Need for Trustworthy Systems	225
Enable Trust in Hardware	226
Secure Hardware for Strong Cryptography	227
Chip Guard for Hardware Integrity	228
Enable Trust in Software	230
Secure Boot	230
Secure iPXE – Secure Boot Over the Network	231
Establish and Maintain Trust at Steady State	231
SELinux	232
SELinux Policy	232
SELinux Mode	233
Role of the SELinux Policy in Boot Process	233
Secure Install	233
RPM Signing and Validation	233
Third-Party RPMs	234
SSD Encryption	234
DM-Crypt	234

AES-NI Support	235
CryptSetup	236
Encrypted Logical Volume	236
SSD Binding	237
Data Zeroization	237
Boot Integrity and Trust Visibility	237
Secure gRPC	244
Integrity Measurement Architecture (IMA)	244
IMA Audit	245
IMA Policy	246
Verifying the IMA Appraisal “enforce” Mode	246
IMA Signatures	246
How Trustworthiness is Implemented	247
Understanding Key Concepts in Security	248



CHAPTER 1

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 1](#)

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 54](#)” section.

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 submenu.

- 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 submenu. 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 52.

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.

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	<p>With this feature, when you key in a password or secret, it is not displayed on the screen. This enhances security.</p> <p>The feature is enabled by default. The following options are added to the username command:</p> <ul style="list-style-type: none"> • 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 6](#) section.

Configuration Example

Directly configuring a Type 8 encrypted password:

```
Router(config)# username demo8
Router(config-un)#secret 8 $8$dsYGNam3K1SIJO$7nv/35M/qr6t.dVc7UY9zrJDWRVqncHub1PE9U1MQFs
```

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.HsgL6x1l10RxxkOSSvyQYwucySct7qFm4v7pqCxxkKM
```

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 $8$mIcSk/Ae5Qu/5k$RJdI3SQ8B4iP7rdxxQvVlJVeRHSubZzcgaLYxjg36s
```

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)# commit
```

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 49](#)
- [Type 10 Password, on page 49](#)

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$9vJidvTEgkAPU$3CLLEi/F.E4v/Hi.UaqLwX8UsSEr9ApG6c5pzhMjnztcw4jObAQ7meAwyhu5VM/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$9UvJidvsTEgkAPU$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 49](#)
- [Backward Compatibility for Password Types, on page 11](#)

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 50](#)

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 or 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.wlJvTPO7IbZry.DxHrE3SN2BBzBJe0
Router(config-un)#commit
```

Running Configuration

```
username user1
policy test-policy1
secret 10
$6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZLhoHd7TicR4mOo8IIVriYCGAKW0A.wlJvTPO7IbZry.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.AWvMKGEz5TLvZ070Xj6DgeOqLoQKI f7XJxKayViFJNateZ0no6gO6DbbXn4bBo/Dlqitro3j1sS40
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 poll
lifetime minutes 1
upper-case 1
restrict-old-count 2
!

username lab2
group root-lr
(2) policy poll
(3) secret 10
$6$gphqA0RfBXOn6A0.$wRwWG110TIpHPdVQ66fUi IM5P46ggoGMGgFuaZd0LD2DLFYD1DPaRyXQLi8Izjb49tC7H7tKTLrc1.GELFpiK.

password 7 1533292F200F2D
!
```

Related Topics

- [Password Policy for User Secret, on page 53](#)

Associated Commands

- **aaa password-policy**
- **policy**
- **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/RP0/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**

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/RP0/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/RP0/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 mode commands.
- 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/RP0/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/RP0/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/RP0/CPU0:router(config)# line console
```

Enters line template configuration mode.

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

Example:

```
RP/0/RP0/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 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/RP0/CPU0:router(config)# line console
```

Enters line template configuration mode.

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

Example:

```
RP/0/RP0/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/RP0/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/RP0/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
```

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 chassis 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-1r:** 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.	<pre>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</pre>
Admin-r/w/x/d	Admin-r	Read only commands on Sysadmin VM	<pre>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</pre>

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	<pre>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#</pre>

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 31](#).
- 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).

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 script 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 11](#).

- 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 12](#).

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 5](#).

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 14](#).

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 cannot 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 2: 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.

Model-based AAA

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>

```

NACM Configuration Management and Persistence

The NACM configuration can be modified using NETCONF or RESTCONF. In order for a user to be able to access 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.

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>

```

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>

```

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>
      <rule>
        <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>
      <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.

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.

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
```

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
```



CHAPTER 2

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 65](#)

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

The following prerequisites are required to implement CA interoperability:

- 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.
- You must install and activate the Package Installation Envelope (PIE) for the security software.

For detailed information about optional PIE installation, refer to the *System Management Guide*.

From Cisco IOS XR Software Release 7.0.1 and later, you need not install the PIE, because the functionality is available in the base image itself.

Whereas, you must still install the PIE for these 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

- You need to have a CA available to your network before you configure this interoperability feature. The CA must support Cisco Systems PKI protocol, the simple certificate enrollment protocol (SCEP) (formerly called certificate enrollment protocol [CEP]).

Restrictions for Implementing Certification Authority

- The software does not support CA server public keys greater than 2048 bits.

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.

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 **crypto key generate rsa [usage keys | general-keys] [keypair-label]**

Example:

```
RP/0/RP0/CPU0:router# crypto key generate 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 **crypto key zeroize rsa [keypair-label]**

Example:

```
RP/0/RP0/CPU0:router# crypto key zeroize rsa key1
```

(Optional) Deletes all RSAs from the router.

- Under certain circumstances, you may want to delete all RSA keys from your 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 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.

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.

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 DF8ADB7F 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.

Prerequisites for CA Trust Pool Management

Restrictions for CA trust pool management

- Device certificates that use CA certificates cannot be enrolled in a CA trust pool.
- Starting with Cisco IOS XR software version 7.3.4, the server certificates (leaf certificates) in the router must have a Fully Qualified Domain Name (FQDN) in the Common Name (CN) field.
- To add an IP address in the Subject Alternate Name (SAN) field of server certificates, add the extension type as IP address in the certificate. If the IP address extension type configuration isn't available, use the [crypto ca fqdn-check ip-address allow](#) command for the router to validate the IP address in the SAN field successfully.

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 76](#) 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/RSP0RP0/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/RSP0RP0/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/RSP0RP0/CPU0:IMC0#show crypto ca trustpool	Displays the CA trust pool certificates of the router in a verbose format.

	Command or Action	Purpose
	<pre>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</pre>	

Configuring Optional Trustpool Policy Parameters

Procedure

	Command or Action	Purpose
Step 1	<p>configure</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# configure</pre>	Enters mode.
Step 2	<p>crypto ca trustpool policy</p> <p>Example:</p> <pre>RP/0/RSP0RP0/CPU0:IMC0(config)#crypto ca trustpool policy RP/0/RSP0/CPU0:IMC0(config-trustpool)#</pre>	Enters ca-trustpool configuration mode where commands can be accessed to configure CA trustpool policy parameters.

	Command or Action	Purpose
Step 3	cabundle url URL Example: RP/0/RSP0RP0/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/RSP0RP0/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/RSP0RP0/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 notification 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-label]
```

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 71](#).

```
Router# crypto ca authenticate [trustpoint-name]
```

```
Router# crypto ca enroll [trustpoint-name]
```

For example,

```
Router# crypto ca authenticate myca
Router# crypto ca enroll myca
```

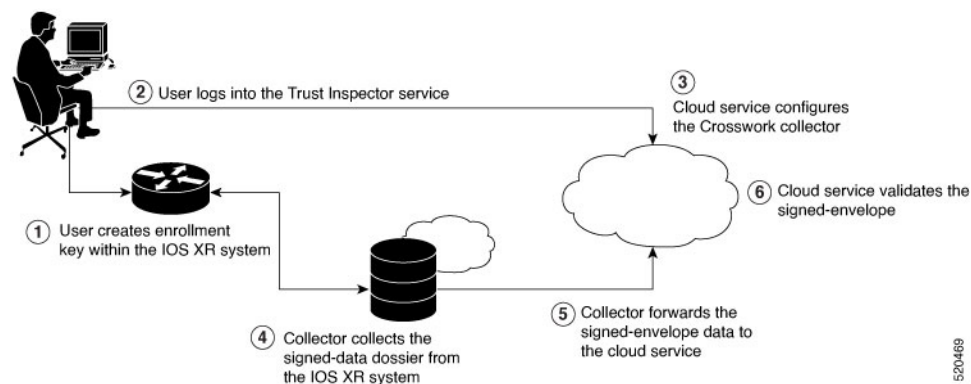
Integrating Cisco IOS XR and Crosswork Trust Insights

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 84](#)
- [Generate System Trust Point for the Leaf and Root Certificate, on page 85](#)
- [Generate Root and Leaf Certificates, on page 86](#)
- [Collect Data Dossier, on page 88](#)

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 nistp521).

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 78](#).

When you see the certificate expiry notification, we recommend you to regenerate the certificate, see [Regenerate the Certificate, on page 80](#).

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 3: Feature History Table

Feature Name	Release Information	Description
Collect Filesystem Inventory	Release 7.3.1	<p>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.</p> <p>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.</p>

Feature Name	Release Information	Description
IMA Optimization	Release 7.3.1	<p>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:</p> <ul style="list-style-type: none"> • 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.

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
NCS540
```

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.



Note Data about System admin, Host, and LC-specific files are not monitored.

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 `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
```

Incremental Integrity Measurement Architecture

With incremental Integrity Measurement 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 81](#) section for configuration steps of each task.

Verify Authenticity of RPM Packages Using Fingerprint

Table 4: Feature History Table

Feature Name	Release Information	Description
Verify Authenticity of RPM Packages Using Fingerprint	Release 7.3.1	<p>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.</p> <p>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.</p>

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>
      </active>
    </packages>
  </install>
</data>
----- 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 5: Feature History Table

Feature Name	Release Information	Feature Description
Support for Ed25519 Public-Key Signature System	Release 7.3.1	<p>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.</p> <p>Commands introduced for this feature are:</p> <p>crypto key generate ed25519</p> <p>crypto key zeroize ed25519</p> <p>show crypto key mypubkey ed25519</p> <p>Commands modified for this feature are:</p> <p>ca-keypair</p> <p>keypair</p>



Note Only the following variants of the Cisco NCS 540 routers support the Ed25519 Public-Key Signature System:

- N540-28Z4C-SYS-A/D
- N540X-16Z4G8Q2C-A/D
- N540-12Z20G-SYS-A/D
- N540X-12Z16G-SYS-A/D
- N540X-6Z18G-SYS-A/D
- N540X-8Z16G-SYS-A/D

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 9EFEC3C3 46A129B2 FAAA12DD EE9D0351
```

Related Topics

- [Support for Ed25519 Public-Key Signature System, on page 94](#)
- [Integrate Cisco IOS XR with Cisco Crosswork Trust Insights using Ed25519, on page 95](#)

Associated Commands

- **crypto key generate ed25519**
- **crypto key zeroize ed25519**
- **show crypto key mypubkey ed25519**

Integrate Cisco IOS XR with Cisco Crosswork Trust Insights using Ed25519

Configuration Example

This section shows how to generate the system trustpoint, and the root and leaf certificates using the Ed25519 signature algorithm, as part of integrating Cisco IOS XR with Cisco Crosswork Trust Insights.

```

Router#configure
Router(config)#domain name domain1
Router(config)#crypto ca trustpoint system-trustpoint
Router(config-trustp)#keypair ed25519 system-enroll-key
Router(config-trustp)#ca-keypair ed25519 system-root-key
Router(config-trustp)#commit

/* Generate root and leaf certificates */
Router#crypto ca authenticate system-trustpoint
Router#crypto ca enroll system-trustpoint

```

Running Configuration

```

config
domain name domain1
crypto ca trustpoint system-trustpoint
  keypair ed25519 system-enroll-key
  ca-keypair ed25519 system-root-key
!

```

For the complete integration procedure, see, [Integrating Cisco IOS XR and Crosswork Trust Insights, on page 81](#).

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 3

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 99](#)

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 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 **show key chain** *key-chain-name*

Example:

```
RP/0/RP0/CPU0:router# show key chain isis-keys
```

(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.

Example

The following example shows how to configure keychain management:

```
configure
key chain isis-keys
accept-tolerance infinite
key 8
key-string mykey9labcd
cryptographic-algorithm MD5
send-lifetime 1:00:00 june 29 2006 infinite
accept-lifetime 1:00:00 june 29 2006 infinite
end

Uncommitted changes found, commit them? [yes]: yes

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]
```

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/RP0/CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **accept-tolerance** *value* [**infinite**]

Example:

```
RP/0/RP0/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 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 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/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
```

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 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 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/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

key-string [**clear** | **password**] *key-string-text*

Example:

```
RP/0/RP0/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 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/RP0/CPU0:router(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/RP0/CPU0:router(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 4

Implementing Type 6 Password Encryption

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 109

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.



Note Follow the first two steps for all Type 6 password encryption scenarios. The third step, *Creating BGP Sessions*, is specific to BGP. To enable Type 6 password encryption for OSPF, IS-IS, or other protocol sessions (the final step), refer the respective configuration guide. For MACsec authentication, refer the *Configure MACsec* chapter.

Enabling Type6 Feature and Creating a Primary Key (Type 6 Server)

The primary key 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.

Configuration

```
/* Enter the primary key details */
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

/* 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. Hence, it is recommended to perform the primary key update operation during a maintenance window, and not while the live session is active. Else, you might experience session flaps due to these configuration changes.

The primary key is not saved to the running configuration, but the changes are persistent across reloads. Please note that the primary key update cannot be rolled back.

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**. It indicates 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 update 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 type6_password
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) Successfully
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: type6_password -
  Key 1 -- text "606745575e656546435a4c4a47694647434253554f49414a4f59655a486950566"
  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

```
/* 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, *type6_password* in this scenario).

Ensure that you use the same session and keychain for all routers (R1 and R2 in this case).

```
R1 (config-bgp)# session-group bgp-type6-session keychain type6_password
R1 (config-bgp)# neighbor 10.1.1.11 remote-as 65537
R1 (config-bgp)# commit

/* Create BGP session on Router2 */
R2 (config)# router bgp 65537
R2 (config-bgp)# session-group bgp-type6-session keychain type6_password
R2 (config-bgp)# neighbor 10.1.1.1 remote-as 65537
R2 (config-bgp)# commit
```

Verification

Verify that the BGP NBR state is in the *Established* state, on R1 and R2.

```
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 5

Understanding MACSec Encryption

Security breaches can occur at any layer of the OSI model. At Layer 2, some of the common breaches are MAC address spoofing, ARP spoofing, Denial of Service (DoS) attacks against a DHCP server, and VLAN hopping.

MACSec secures data on physical media, making it impossible for data to be compromised at higher layers. As a result, MACSec encryption takes priority over any other encryption method such as IPsec and SSL at higher layers. MACSec is configured on the Customer Edge (CE) router interfaces that connect to Provider Edge (PE) routers and on all the provider router interfaces.

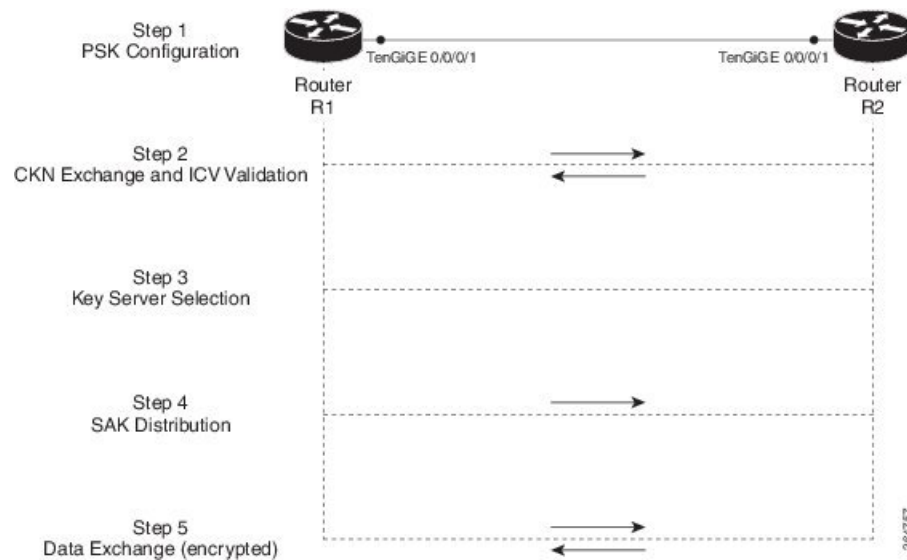
- [MKA Authentication Process, on page 115](#)
- [MACsec Frame Format, on page 116](#)
- [Advantages of Using MACsec Encryption, on page 117](#)
- [Hardware Support for MACSec, on page 117](#)
- [MACsec PSK, on page 118](#)
- [Fallback PSK, on page 118](#)
- [Configuring and Verifying MACsec Encryption , on page 119](#)
- [Creating a MACsec Keychain, on page 119](#)
- [Creating a User-Defined MACsec Policy, on page 122](#)
- [Applying MACsec Configuration on an Interface, on page 124](#)
- [MACsec Policy Exceptions, on page 126](#)
- [Verifying MACsec Encryption on IOS XR, on page 127](#)
- [Verifying MACsec Encryption on the Router, on page 133](#)

MKA Authentication Process

MACsec provides the secure MAC Service on a frame-by-frame basis, using GCM-AES algorithm. MACsec uses the MACsec Key Agreement protocol (MKA) to exchange session keys, and manage encryption keys.

The MACsec encryption process is illustrated in the following figure and description.

Figure 1: MKA Encryption Process



Step 1: When a link is first established between two routers, they become peers. Mutual peer authentication takes place by configuring a Pre-shared Key (PSK).

Step 2: On successful peer authentication, a connectivity association is formed between the peers, and a secure Connectivity Association Key Name (CKN) is exchanged. After the exchange, the MKA ICV is validated with a Connectivity Association Key (CAK), which is effectively a secret key.

Step 3: A key server is selected between the routers, based on the configured key server priority. Lower the priority value, higher the preference for the router to become the key server. If no value is configured, the default value of 16 is taken to be the key server priority value for the router. Lowest priority value configures that router as the key server, while the other router functions as a key client. The following rules apply to key server selection:

- Numerically lower values of key server priority and SCI are accorded the highest preference.
- Each router selects a peer advertising the highest preference as its key server provided that peer has not selected another router as its key server or is not willing to function as the key server.
- In the event of a tie for highest preferred key server, the router with the highest priority SCI is chosen as key server (KS).

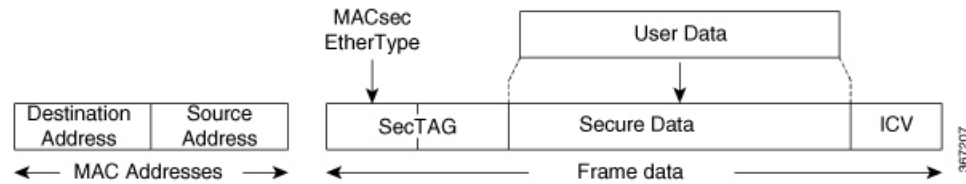
Step 4: A security association is formed between the peers. The key server generates and distributes the Secure Association Key (SAK) to the key client (peer). Each secure channel is supported by an overlapped sequence of Security Associations (SA). Each SA uses a new Secure Association Key (SAK).

Step 5: Encrypted data is exchanged between the peers.

MACsec Frame Format

The MACsec header in a frame consists of three components as illustrated in the following figure.

Figure 2: MACsec Frame Format



- **SecTAG:** The security tag is 8-16 bytes in length and identifies the SAK to be used for the frame. With Secure Channel Identifier (SCI) encoding, the security tag is 16 bytes in length, and without the encoding, 8 bytes in length (SCI encoding is optional). The security tag also provides replay protection when frames are received out of sequence.
- **Secure Data:** This is the data in the frame that is encrypted using MACsec and can be 2 or more octets in length.
- **ICV:** The ICV provides the integrity check for the frame and is usually 8-16 bytes in length, depending on the cipher suite. Frames that do not match the expected ICV are dropped at the port.

Advantages of Using MACsec Encryption

- **Data Integrity Check:** Integrity check value (ICV) is used to perform integrity check. The ICV is sent with the protected data unit and is recalculated and compared by the receiver to detect data modification.
- **Data Encryption:** Enables a port to encrypt outbound frames and decrypt MACsec-encrypted inbound frames.
- **Replay Protection:** When frames are transmitted through the network, there is a strong possibility of frames getting out of the ordered sequence. MACsec provides a configurable window that accepts a specified number of out-of-sequence frames.
- **Support for Clear Traffic:** If configured accordingly, data that is not encrypted is allowed to transit through the port.

Hardware Support for MACSec



Note MACSec is supported only on the QSFP28 or SFP28 ports (in 10G to 100G speeds, and not 1G speed).

Release History for MACSec Hardware Support

Table 6: Release History for MACSec Hardware Support

Release	Description
Release 7.3.1	This release introduces the support for MACSec on the following NCS 540 router variants: <ul style="list-style-type: none"> • N540-ACC-SYS • N540-24Z8Q2C-SYS
Release 7.3.1	MACsec is supported on SFP28 and QSFP28 ports on the following NCS540 variants: <ul style="list-style-type: none"> • N540-ACC-SYS • N540X-ACC-SYS (Premium) • N540-24Z8Q2C-SYS

MACsec PSK

A pre-shared key includes a connectivity association key name (CKN) and a connectivity association key (CAK). A pre-shared key is exchanged between two devices at each end of a point-to-point link to enable MACsec using static CAK security mode. The MACsec Key Agreement (MKA) protocol is enabled after the pre-shared keys are successfully verified and exchanged. The pre-shared keys, the CKN and CAK, must match on both ends of a link.

For more information on MACsec PSK configuration, see [Step 3, on page 125](#) of the [Applying MACsec Configuration on an Interface, on page 124](#) section.

Fallback PSK

Fallback is a session recovery mechanism when primary PSK fails to bring up secured MKA session. It ensures that a PSK is always available to perform MACsec encryption and decryption.

- In CAK rollover of primary keys, if latest active keys are mismatched, system performs a hitless rollover from current active key to fallback key, provided the fallback keys match.
- If a session is up with fallback, and primary latest active key configuration mismatches are rectified between peers, system performs a hitless rollover from fallback to primary latest active key.



Note A valid Fallback PSK (CKN and CAK) must be configured with infinite lifetime. If the fallback PSK is configured with CAK mismatch, the only recovery mechanism is to push a new set of PSK configurations (both on fallback PSK keychain and primary PSK chain in that order) on all the association members.

The following is a sample syslog for session secured with fallback PSK:


```
%L2-MKA-5-SESSION_SECURED_WITH_FALLBACK_PSK : (Hu0/1/0/0) MKA session secured, CKN:ABCD
```

For more information on MACsec fallback PSK configuration, see [Step 3, on page 125](#) of the [Applying MACsec Configuration on an Interface, on page 124](#) section.

Configuring and Verifying MACsec Encryption

MACsec can be configured on physical ethernet interfaces or member links of the interface bundles, as explained in this section.

The following section describes procedures for configuring and verifying MACsec configuration in the described deployment modes.

Prior to configuring MACsec on a router interface the MACsec keychain must be defined. If you apply the MACsec keychain on the router without specifying a MACsec policy, the default policy is applied. A default MACsec policy is pre-configured with default values. If you need to change any of the pre-configured values, create a different MACsec policy.

Configuring MACsec involves the following steps:

1. Creating a MACsec keychain
2. Creating a user-defined MACsec policy
3. Applying MACsec configuration on physical interfaces

Creating a MACsec Keychain

A MACsec keychain is a collection of keys used to authenticate peers needing to exchange encrypted information. While creating a keychain, we define the key(s), key string with password, the cryptographic algorithm, and the key lifetime.

MACsec Keychain Keyword	Description
Key	The MACsec key or the CKN can be up to 64 characters in length. The key must be of an even number of characters. Entering an odd number of characters will exit the MACsec configuration mode.
Key-string	The MACsec key-string or the CAK can be either 32 characters or 64 characters in length (32 for AES-128, 64 for AES-256).
Lifetime	This field specifies the validity period of a key. It includes a start time, and an expiry time. We recommend you to set the value for expiry time as <i>infinite</i> .

Guidelines for Configuring MACsec Keychain

MACsec keychain management has the following configuration guidelines:

- To establish MKA session, ensure that the MACsec key (CKN) and key-string (CAK) match at both ends.
- MKA protocol uses the latest active key available in the Keychain. This key has the latest Start Time from the existing set of currently active keys. You can verify the values using the **show key chain keychain-name** command.
- Deletion or expiry of current active key brings down the MKA session resulting in traffic hit. We recommend you to configure the keys with infinite lifetime. If fallback is configured, traffic is safeguarded using fallback on expiry or deletion of primary-keychain active key.
- To achieve successful key rollover (CAK-rollover), the new key should be configured such that it is the latest active key, and kicks-in before the current key expires.
- We recommend an overlap of at least one minute for hitless CAK rollover from current key to new key.
- Start time and Expiry time can be configured with future time stamps, which allows bulk configuration for daily CAK rotation without any intervention of management agent.

Procedure

Step 1 Enter the global configuration mode and provide a name for the MACsec keychain; for example, mac_chain.

Example:

```
RP/0/RP0/CPU0:router(config)# key chain mac_chain
```

Step 2 Enter the MACsec mode.

Example:

```
RP/0/RP0/CPU0:router(config-mac_chain)#macsec
```

Step 3 Provide a name for the MACsec key.

The key can be up to 64 characters in length. The key must be of an even number of characters. Entering an odd number of characters will exit the MACsec configuration mode.

Example:

```
RP/0/RP0/CPU0:router(config-mac_chain-MacSec)#key 1234abcd5678
```

You can also configure a fall-back pre-shared key (PSK) to ensure that a PSK is always available to perform MACsec encryption and decryption. The fallback PSK along with the primary PSK ensures that the session remains active even if the primary PSK is mismatched or there is no active key for the primary PSK.

The configured key is the CKN that is exchanged between the peers.

Note If you are configuring MACsec to interoperate with a MACsec server that is running software prior to Cisco IOS XR Release 6.1.3, then ensure that the MACsec key length is of 64 characters. You can add extra zero characters to the MACsec key so that the length of 64-characters is achieved. If the key length is lesser than 64 characters, authentication will fail.

This completes the configuration of the MACsec keychain.

Creating a User-Defined MACsec Policy

Procedure

Step 1 Enter the global configuration mode, and enter a name (mac_policy) for the MACsec policy.

Example:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# macsec-policy mac_policy
```

Step 2 Configure the cipher suite to be used for MACsec encryption.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# cipher-suite GCM-AES-XPN-256
RP/0/RP0/CPU0:router(config-mac_policy)#GCM-AES-128
GCM-AES-256
GCM-AES-XPN-128
GCM-AES-XPN-256
```

Note In this example, we have used the GCM-AES-XPN-256 encryption algorithm. A 256-bit encryption algorithm uses a larger key that requires more rounds of hacking to be cracked. 256-bit algorithms provide better security against large mass security attacks, and include the security provided by 128-bit algorithms. Extended Packet Numbering (XPN) is used to reduce the number of key rollovers while data is sent over high speed links. It is therefore highly recommended to use GCM-AES-XPN-256 encryption algorithm for higher data ports.

Step 3 Configure the confidentiality offset for MACsec encryption.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# conf-offset CONF-OFFSET-30
```

Note We recommend to change the offset value of the **conf-offset** <offset_value> command (MACsec encryption command) in only when the port is in **admin down** state (that is, when the interface is shut down). Changing the offset value otherwise may result in traffic loss.

Step 4 Enter the key server priority.

You can enter a value between 0-255. Lower the value, higher the preference to be selected as the key server.

In this example, a value of 0 configures the router as the key server, while the other router functions as a key client. The key server generates and maintains the SAK between the two routers. The default key server priority value is 16.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# key-server-priority 0
```

Step 5 Configure the security policy parameters, either Must-Secure or Should-Secure.

Must-Secure: Must-Secure imposes only MACsec encrypted traffic to flow. Hence, until MKA session is not secured, traffic will be dropped.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# security-policy must-secure
```

Should-Secure: Should-Secure allows unencrypted traffic to flow until MKA session is secured. After the MKA session is secured, Should-Secure policy imposes only encrypted traffic to flow.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# security-policy should-secure
```

Table 7: MACsec Security Policies

MKA		Secured MKA Session	Unsecured MKA Session
Security Policy	Must-secure	Encrypted traffic	Traffic drop (no Tx and no Rx)
	Should-secure	Encrypted traffic	Plain text or unencrypted traffic

Step 6

Configure data delay protection under MACsec policy.

Data delay protection allows MKA participants to ensure that the data frames protected by MACsec are not delayed by more than 2 seconds. Each SecY uses MKA to communicate the lowest PN used for transmission with the SAK within two seconds. Traffic delayed longer than 2 seconds are rejected by the interfaces enabled with delay protection.

By default, the data delay protection feature is disabled. Configuring the **delay-protection** command under MACsec-policy attached to MACsec interface will enable the data delay protection feature on that interface.

Example:

```
RP/0/RP0/CPU0:router# configure terminal
RP/0/RP0/CPU0:router(config)# macsec-policy mp1
RP/0/RP0/CPU0:router(config-macsec-policy)# delay-protection
RP/0/RP0/CPU0:router(config-macsec-policy)# commit
```

Verification:

The following show command output verifies that the data delay protection feature is enabled.

Example:

```
RP/0/RP0/CPU0:router# show macsec mka session interface GigabitEthernet 0/1/0/1 detail
MKA Policy Name       : mp1
Key Server Priority   : 16
Delay Protection      : TRUE
Replay Window Size    : 64
Confidentiality Offset : 0
Algorithm Agility     : 80C201
SAK Cipher Suite      : (NONE)
MACsec Capability     : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired        : YES
```

Step 7

Configure the replay protection window size.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# window-size 64
```

This dictates the maximum out-of-sequence frames that are accepted. You can configure a value between 0 and 1024.

Step 8 Configure the ICV for the frame arriving on the port.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# include-icv-indicator
```

This parameter configures inclusion of the optional ICV Indicator as part of the transmitted MACsec Key Agreement PDU (MKPDU). This configuration is necessary for MACsec to interoperate with routers that run software prior to IOS XR version 6.1.3. This configuration is also important in a service provider WAN setup where MACsec interoperates with other vendor MACsec implementations that expect ICV indicator to be present in the MKPDU.

Step 9 Commit your configuration and exit the global configuration mode.

Example:

```
RP/0/RP0/CPU0:router(config-mac_policy)# exit
RP/0/RP0/CPU0:router(config)# commit
RP/0/RP0/CPU0:router(config)# exit
```

Step 10 Confirm the MACsec policy configuration.

Example:

```
RP/0/RP0/CPU0:router# show running-config macsec-policy

macsec-policy mac_policy
conf-offset CONF-OFFSET-30
security-policy must-secure
window-size 64
cipher-suite GCM-AES-XPB-256
key-server-priority 0
include-icv-indicator
```

This completes the configuration of the MACsec policy.



Note

- Small packets might be dropped when Data Delay Protection (DDP) is enabled on many MACsec enabled interfaces of a scaled setup. To avoid this, enable DDP only on the interfaces which are absolutely necessary.
-

Applying MACsec Configuration on an Interface

The MACsec service configuration is applied to the host-facing interface of a CE router.

Guidelines for MACsec Interface Configuration

Following are the guidelines for configuring MACsec interface:

- Configure different keychains for primary and fallback PSKs.

- We do not recommend to update both primary and fallback PSKs simultaneously, because fallback PSK is intended to recover MACsec session on primary key mismatch.
- When using MACsec, we recommend you adjust the maximum transmission unit (MTU) of an interface to accommodate the MACsec overhead. Configuring MTU value on an interface allows protocols to do MTU negotiation including MACsec overhead. For instance, if the default MTU is 1514 bytes, configure the MTU to 1546 bytes (1514 + 32).
- The minimum MTU for IS-IS protocol on the MACsec interface is 1546 bytes.
- For enabling MACsec on bundle members :
 - We recommend configuring the maximum possible MTU on the bundle interface.
 - The MTU configurations must account for the maximum packet size of the protocols running on the bundle interface and 32 bytes of MACsec overhead.
 - For IS-IS protocol running on the bundle interface, hello-padding must be disabled.

Before you begin

The requirements for enabling IS-IS on a MACsec-configured interface are:

- The minimum MTU for IS-IS to work with MACsec is 1546.
- On IS-IS instances that are running on bundle interface with MACsec enabled members, ensure that `hello-padding disable` is configured under the IS-IS configuration.

Procedure

Step 1 Enter the global configuration mode.

Example:

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

Step 2 Enter the interface configuration mode.

Example:

```
RP/0/RP0/CPU0:router(config)# interface Te0/3/0/1/4
```

Step 3 Apply the MACsec configuration on an interface.

MACsec PSK Configuration

To apply MACsec PSK configuration on an interface, use the following command.

Example:

```
RP/0/RP0/CPU0:router(config-if)# macsec psk-keychain mac_chain policy mac_policy  
RP/0/RP0/CPU0:router(config-if)# exit
```

To apply MACsec configuration on a physical interface without the MACsec policy, use the following command.

Example:

```
RP/0/RP0/CPU0:router(config-if)# macsec psk-keychain script_key_chain2
RP/0/RP0/CPU0:router(config-if)# exit
```

MACsec Fallback PSK Configuration

To apply MACsec configuration on a physical interface with a fallback PSK, use the following command.

Example:

```
RP/0/RP0/CPU0:router(config-if)# macsec psk-keychain mac_chain fallback-psk-keychain
fallback_mac_chain policy mac_policy
RP/0/RP0/CPU0:router(config-if)# exit
```

It is optional to configure a fallback PSK. If a fallback PSK is configured, the fallback PSK along with the primary PSK ensures that the session remains active even if the primary PSK is mismatched, or there is no active key for the primary PSK.

Step 4 Commit your configuration.

Example:

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

MACsec Policy Exceptions

By default, the MACsec security policy uses **must-secure** option, that mandates data encryption. Hence, the packets cannot be sent in clear-text format. To optionally bypass the MACsec encryption or decryption for Link Aggregation Control Protocol (LACP) packets, and to send the packets in clear-text format, use the **policy-exception lacp-in-clear** command in macsec-policy configuration mode. This functionality is beneficial in scenarios such as, in a network topology with three nodes, where bundles are terminated at the middle node, whereas MACsec is terminated at the end nodes.

This MACsec policy exception is also beneficial in interoperability scenarios where the node at the other end expects the data packets to be in clear text.

How to Create MACsec Policy Exception

Configuration Example

Using the **policy-exception** command:

```
Router#configure
Router(config)#macsec-policy test-macsec-policy
Router(config-macsec-policy)#policy-exception lacp-in-clear
Router(config-macsec-policy)#commit
```

Running Configuration

With the **policy-exception** command:

```
Router#show run macsec-policy test-macsec-policy
macsec-policy test-macsec-policy
```



```

policy-exception lacp-in-clear
security-policy should-secure
include-icv-indicator
sak-rekey-interval seconds 120
!

```

Associated Commands

- **policy-exception lacp-in-clear**

Verifying MACsec Encryption on IOS XR

MACsec encryption on IOS XR can be verified by running relevant commands in the Privileged Executive Mode. The verification steps are the same for MACsec encryption on L2VPN or L3VPN network.

To verify if MACsec encryption has been correctly configured, follow these steps.

Procedure

Step 1 Verify the MACsec policy configuration.

Example:

```
RP/0/RP0/CPU0:router#show macsec policy mac_policy
```

```

=====
Policy      Cipher      Key-Svr      Window  Conf
name        Suite        Priority      Size    Offset
=====
mac_policy  GCM-AES-XP  0             64      30
            N-256

```

If the values you see are different from the ones you configured, then check your configuration by running the **show run macsec-policy** command.

Step 2 Verify the MACsec configuration on the respective interface.

You can verify the MACsec encryption on the configured interface bundle (MPLS network).

Example:

```
RP/0/RP0/CPU0:router#show macsec mka summary
```

```
NODE: node0_0_CPU0
```

```

=====
Interface   Status   Cipher Suite      KeyChain
=====
Fo0/0/0/1/0 Secured  GCM-AES-XP  mac_chain
            N-256
Total MACSec Sessions : 1

```

```
Secured Sessions : 1
Pending Sessions : 0
```

```
RP/0/RP0/CPU0:router# show macsec mka session interface Fo0/0/0/1/0
```

```
=====
Interface           Local-TxSCI           # Peers           Status           Key-Server
=====
Fo0/0/0/1/0         d46d.5023.3709/0001   1                 Secured          YES
=====
```

The **Status** field in the output confirms that the respective interface is **Secured**. If MACsec encryption is not successfully configured, you will see a status such as **Pending** or **Init**.

Run the **show run macsec-policy** command in the privileged executive mode to troubleshoot the configuration entered.

Step 3

Verify whether the interface of the router is peering with its neighbor after MACsec configuration. The MACsec PSK validation detects inconsistency or mismatch of primary and fallback keys (CAK) being used by MKA, allowing operators to rectify the mismatch.

Example:

The **show macsec mka session interface *interface* detail** command carries the Peer Validation status in the **Peer CAK** field. The values of this field can be either *Match* or *Mismatch*.

The following show command output verifies that the primary and fallback keys (CAK) are matched on both peer ends.

```
• RP/0/RP0/CPU0:router#show macsec mka session detail
Peers Status:
  Last Tx MKPDU       : 2017 Sep 02 11:24:52.369
  Peer Count          : 1
  RxSCI               : 008A960060900001
  MI                  : C2213E81C953A202C08DB999
  Peer CAK            : Match
  Latest Rx MKPDU     : 2017 Sep 02 11:24:53.360
Fallback Data:
  CKN                 : ABCD
  MI                   : 84E724B4BA07CE414FEA84EF
  MN                   : 8
Peers Status:
  Last Tx MKPDU       : 2017 Sep 02 11:24:52.369
  Peer Count          : 1
  RxSCI               : 008A960060900001
  MI                  : D2B902453F90389BD3385F84
  Peer CAK            : Match
  Latest Rx MKPDU     : 2017 Sep 02 11:24:53.360
```

• Syslog

```
%L2-MKA-6-MKPDU_ICV_SUCCESS: (Hu0/5/0/1), ICV verification success for
RxSCI(008a.9600.6090/0001), CKN(1000)
%L2-MKA-6-FALLBACK_PSK_MKPDU_ICV_SUCCESS: (Hu0/5/0/1), ICV verification success for
RxSCI(008a.9600.6090/0001), CKN(FFFF)
```

The following show command output verifies that the primary and fallback keys (CAK) are mismatched on both peer ends.

```
• RP/0/RP0/CPU0:router#show macsec mka session detail
Peers Status:
  Last Tx MKPDU       : 2017 Sep 02 11:24:52.369
  Peer Count          : 1
  RxSCI               : 008A960060900001
  MI                  : C2213E81C953A202C08DB999
```



```

Latest SAK Status      : Rx & Tx
Latest SAK AN         : 0
Latest SAK KI (KN)    : E3D39135831AFBCDA7AA9DBB00000001 (1)
Old SAK Status        : FIRST-SAK
Old SAK AN            : 0
Old SAK KI (KN)       : FIRST-SAK (0)

SAK Transmit Wait Time : 0s (Not waiting for any peers to respond)
SAK Retire Time        : 0s (No Old SAK to retire)
Time to SAK Rekey     : NA
MKA Policy Name       : *DEFAULT POLICY*
Key Server Priority    : 16
Replay Window Size    : 64
Confidentiality Offset : 0
Algorithm Agility     : 80C201
SAK Cipher Suite      : 0080C20001000004 (GCM-AES-XPB-256)
MACsec Capability     : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired        : YES

# of MACsec Capable Live Peers      : 1
# of MACsec Capable Live Peers Responded : 0

Live Peer List:
  MI                MN                Rx-SCI (Peer)                SSCI KS-Priority
  -----
RP/0/RP0/CPU0:router#show macsec mka session interface Fo0/0/0/1/0 detail Tue May 18
13:23:29.935 UTC
Tue May 18 13:23:29.935 UTC

MKA Detailed Status for MKA Session
=====
Status: Secured - Secured MKA Session with MACsec

Local Tx-SCI          : 008a.96d6.194c/0001
Local Tx-SSCI         : 2
Interface MAC Address : 008a.96d6.194c
MKA Port Identifier   : 1
Interface Name        : Hu0/2/0/11
CAK Name (CKN)       : 2111
CA Authentication Mode : PRIMARY-PSK
Keychain              : test1
Member Identifier (MI) : 69B39E87B3CBA673401E9891
Message Number (MN)  : 352
Authenticator         : NO
Key Server            : YES
MKA Cipher Suite      : AES-128-CMAC
Configured MACSec Cipher Suite : GCM-AES-XPB-128
Key Distribution Mode  : SAK

Latest SAK Status      : Rx & Tx
Latest SAK AN         : 0
Latest SAK KI (KN)    : 69B39E87B3CBA673401E989100000001 (1)
Old SAK Status        : FIRST-SAK
Old SAK AN            : 0
Old SAK KI (KN)       : FIRST-SAK (0)

SAK Transmit Wait Time : 0s (Not waiting for any peers to respond)
SAK Retire Time        : 0s (No Old SAK to retire)
Time to SAK Rekey     : 456s
Time to exit suspension : NA

MKA Policy Name       : P12
Key Server Priority    : 20
Delay Protection      : TRUE

```

```

Replay Window Size           : 100
Include ICV Indicator        : TRUE
Confidentiality Offset       : 0
Algorithm Agility            : 80C201
SAK Cipher Suite             : 0080C20001000003 (GCM-AES-XPB-128)
MACsec Capability            : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired               : YES

```

```

# of MACsec Capable Live Peers      : 1
# of MACsec Capable Live Peers Responded : 1

```

```

# of MACSec Suspended Peers        : 0

```

Live Peer List:

```

-----
                MI                MN                Rx-SCI                SSCI  KS-Priority
-----
42A78BD6243539E917B8C6B2      290          7061.7bea.1df4/0001      1        20

```

Potential Peer List:

```

-----
                MI                MN                Rx-SCI                SSCI  KS-Priority
-----

```

Suspended Peer List:

```

-----
                Rx-SCI                SSCI
-----

```

Peers Status:

```

Last Tx MKPDU           : 2021 May 18 13:23:29.588
Peer Count               : 1

```

```

RxSCI                   : 70617BEA1DF40001
MI                       : 42A78BD6243539E917B8C6B2
Peer CAK                 : Match
Latest Rx MKPDU         : 2021 May 18 13:23:29.847

```

MKA Detailed Status for MKA Session

```

=====
Status: Active - Marked Peer as Live (Waiting for SAK generation/distribution)

```

```

Local Tx-SCI             : 008a.96d6.194c/0001
Local Tx-SSCI           : 2
Interface MAC Address    : 008a.96d6.194c
MKA Port Identifier     : 1
Interface Name          : Hu0/2/0/11
CAK Name (CKN)         : 2000
CA Authentication Mode  : FALLBACK-PSK
Keychain                : test1f
Member Identifier (MI)  : 8F59AD6021FA3E2D5F9E6231
Message Number (MN)    : 350
Authenticator           : NO
Key Server              : YES
MKA Cipher Suite        : AES-128-CMAC
Configured MACSec Cipher Suite : GCM-AES-XPB-128
Key Distribution Mode    : SAK

Latest SAK Status       : Rx & Tx
Latest SAK AN           : 0
Latest SAK KI (KN)     : 69B39E87B3CBA673401E989100000001 (1)
Old SAK Status          : FIRST-SAK
Old SAK AN              : 0
Old SAK KI (KN)        : FIRST-SAK (0)

```

```

SAK Transmit Wait Time      : 0s (Not waiting for any peers to respond)
SAK Retire Time             : 0s (No Old SAK to retire)
Time to SAK Rekey          : 456s
Time to exit suspension    : NA

MKA Policy Name            : P12
Key Server Priority        : 20
Delay Protection           : TRUE
Replay Window Size        : 100
Include ICV Indicator      : TRUE
Confidentiality Offset     : 0
Algorithm Agility          : 80C201
SAK Cipher Suite           : 0080C20001000003 (GCM-AES-XPB-128)
MACsec Capability          : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired             : YES

# of MACsec Capable Live Peers      : 1
# of MACsec Capable Live Peers Responded : 0

# of MACSec Suspended Peers        : 0

```

Live Peer List:

```

-----
                MI                MN                Rx-SCI                SSCI  KS-Priority
-----
1BB9428C721F6EE3E538C942      288          7061.7bea.1df4/0001      1        20

```

Potential Peer List:

```

-----
                MI                MN                Rx-SCI                SSCI  KS-Priority
-----

```

Suspended Peer List:

```

-----
                Rx-SCI                SSCI
-----

```

Peers Status:

```

Last Tx MKPDU      : 2021 May 18 13:23:29.587
Peer Count         : 1

```

```

RxSCI              : 70617BEA1DF40001
MI                 : 1BB9428C721F6EE3E538C942
Peer CAK           : Match
Latest Rx MKPDU   : 2021 May 18 13:23:29.847

```

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

The **Status** field in the output verifies if the MKA session is secured with MACsec encryption. The output also displays information about the interface and other MACsec parameters.

Step 5 Verify the MACsec session counter statistics.

Example:

```
RP/0/RP0/CPU0:router# show macsec mka statistics interface Fo0/0/0/1/0
```

```

MKA Statistics for Session on interface (Fo0/0/0/1/0)
=====
Reauthentication Attempts.. 0

```

```

CA Statistics
Pairwise CAKs Derived... 0
Pairwise CAK Rekeys..... 0
Group CAKs Generated... 0
Group CAKs Received..... 0

SA Statistics
SAKs Generated..... 3
SAKs Rekeyed..... 2
SAKs Received..... 0
SAK Responses Received.. 3

MKPDU Statistics
MKPDUs Transmitted..... 5425
"Distributed SAK".. 8
"Distributed CAK".. 0
MKPDUs Validated & Rx... 4932
"Distributed SAK".. 0
"Distributed CAK".. 0

MKA IDB Statistics
MKPDUs Tx Success..... 5425
MKPDUs Tx Fail..... 0
MKPDUS Tx Pkt build fail... 0
MKPDUs Rx CA Not found.... 0
MKPDUs Rx Error..... 0
MKPDUs Rx Success..... 4932

MKPDU Failures
MKPDU Rx Validation (ICV)..... 0
MKPDU Rx Bad Peer MN..... 0
MKPDU Rx Non-recent Peerlist MN..... 0
MKPDU Rx Drop SAKUSE, KN mismatch..... 0
MKPDU Rx Drop SAKUSE, Rx Not Set..... 0
MKPDU Rx Drop SAKUSE, Key MI mismatch.. 0
MKPDU Rx Drop SAKUSE, AN Not in Use.... 0
MKPDU Rx Drop SAKUSE, KS Rx/Tx Not Set. 0

SAK Failures
SAK Generation..... 0
Hash Key Generation..... 0
SAK Encryption/Wrap..... 0
SAK Decryption/Unwrap..... 0

```

The counters display the MACsec PDUs transmitted, validated, and received. The output also displays transmission errors, if any.

This completes the verification of MACsec encryption on the IOS-XR.

Verifying MACsec Encryption on the Router

MACsec encryption on the router hardware can be verified by running relevant commands in the Privileged Executive Mode.

To verify if MACsec encryption has been correctly configured, follow these steps.

Procedure

Step 1 Verify the MACsec encryption and hardware interface descriptor block (IDB) information on the interface.

Example:

```
RP/0/RP0/CPU0:router# show macsec ea idb interface Fo0/0/0/1/0
```

```
IDB Details:
if_sname : Fo0/0/0/1/0
if_handle : 0x3480
Replay window size : 64
Local MAC : 00:1d:e5:e9:aa:39
Rx SC Option(s) : Validate-Frames Replay-Protect
Tx SC Option(s) : Protect-Frames Always-Include-SCI
Security Policy : MUST SECURE
Sectag offset : 8
Rx SC 1
Rx SCI : 001de5e9b1bf0019
Peer MAC : 00:1d:e5:e9:b1:bf
Stale : NO
SAK Data
SAK[0] : ***
SAK Len : 32
HashKey[0] : ***
HashKey Len : 16
Conf offset : 30
Cipher Suite : GCM-AES-XPB-256
CtxSalt[0] : 83 c3 7b ad 7b 6f 63 16 09 8f f3 d2
Rx SA Program Req[0]: 2015 Oct 09 15:20:53.082
Rx SA Program Rsp[0]: 2015 Oct 09 15:20:53.092

Tx SC
Tx SCI : 001de5e9aa39001a
Active AN : 0
Old AN : 255
Next PN : 1, 0, 0, 0
SAK Data
SAK[0] : ***
SAK Len : 32
HashKey[0] : ***
HashKey Len : 16
Conf offset : 30
Cipher Suite : GCM-AES-XPB-256
CtxSalt[0] : 83 c3 7b ae 7b 6f 63 16 09 8f f3 d2
Tx SA Program Req[0]: 2015 Oct 09 15:20:55.053
Tx SA Program Rsp[0]: 2015 Oct 09 15:20:55.064
```

The **if_handle** field provides the IDB instance location.

The **Replay window size** field displays the configured window size.

The **Security Policy** field displays the configured security policy.

The **Local Mac** field displays the MAC address of the router.

The **Peer Mac** field displays the MAC address of the peer. This confirms that a peer relationship has been formed between the two routers.

Step 2 Use the IDB handle retrieved from Step 1 to verify the platform hardware information.

Example:

```
RP/0/RP0/CPU0:router# show macsec platform hardware
idb location 0/0/CPU0 | b 3480

if_handle : 0x00003480
NPPort : 099 [0x063]
LdaPort : 016 [0x010] SerdesPort : 000 [0x000]
NetSoftPort : 061 [0x03d] SysSoftPort : 062 [0x03e]
Active AN : 0x00000000 Idle AN : 0x000000ff
Match-All Tx SA : 0x80010001 Match-All Rx SA : 0x00010001
Match-All Tx Flow : 0x80000003 Match-All Rx Flow : 0x00000003
Bypass Tx SA : 0x80000000 Bypass Rx SA : 0x00000000
Tx SA[0] : 0x80020002 Tx Flow[0] : 0x8000000c
Tx SA[1] : 0xffffffff Tx Flow[1] : 0xffffffff
Tx SA[2] : 0xffffffff Tx Flow[2] : 0xffffffff
Tx SA[3] : 0xffffffff Tx Flow[3] : 0xffffffff
Rx SA[0] : 0x00020002 Rx Flow[0] : 0x0000000c
Rx SA[1] : 0xffffffff Rx Flow[1] : 0xffffffff
Rx SA[2] : 0xffffffff Rx Flow[2] : 0xffffffff
Rx SA[3] : 0xffffffff Rx Flow[3] : 0xffffffff
```

- Step 3** Use the Transmitter SA retrieved from Step 2 to verify the MACsec SA information programmed in the hardware.

Example:

```
RP/0/RP0/CPU0:router# show macsec platform hardware sa
0x80020002 interface Fo0/0/0/1/0 location 0/0/CPU0

MACsec HW SA Details:
Action Type : 0x00000003
Direction : Egress
Dest Port : 0x00000000
Conf Offset : 00000030
Drop Type : 0x00000002
Drop NonResvd : 0x00000000
SA In Use : YES
ConfProtect : YES
IncludeSCI : YES
ProtectFrame : YES
UseEs : NO
UseSCB : NO
SCI : 00 1d e5 e9 aa 39 00 05
Replay Window : 64 MacsecCryptoAlgo : 7
Direction : Egress AN : 0
AES Key Len : 256 X-Packet Number : 0x0000000000000000
CtxSalt : f8d88dc3e1c5e6a94ca2299
```

The output displays the details of the encryption, such as the AES key, the Auth key, and other parameters.

- Step 4** Verify the MACsec Secure Channel (SC) information programmed in the hardware.

Example:

```
RP/0/RP0/CPU0:router# show macsec platform hardware msc
interface Fo0/0/0/1/0 location 0/0/CPU0
```

```
MACsec HW Cfg Details:
Mode : 0x5
Counter Clear on Read : 0x0
SA Fail Mask : 0xffff
Global SecFail Mask : 0xffffffff
Latency : 0xff
StaticBypass : 0x0
Should secure : 0x0
Global Frame Validation : 0x2
Ctrl Pkt CC Bypass : 0x1
NonCtrl Pkt CC Bypass : 0x1
Sequence Number Threshold : 0xbffffb8
Sequence Number Threshold 64bit : 0x000002fffffffffd
Non Matching Non Control Pkts Programming
    Untagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
    Tagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
    BadTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
    KayTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
Non Matching Control Pkts Programming
    Untagged : Bypass: 0x1 DestPort : 0x2, DropType : 0xffffffff
    Tagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
    BadTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
    KayTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
```

This completes the verification of MACsec encryption on the router hardware.

This completes the configuration and verification of MACsec encryption.



CHAPTER 6

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 540 Series Routers to prevent unauthorized devices (clients) from gaining access to the network.

Table 8: 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.

- [Restrictions for IEEE 802.1X Port-Based Authentication, on page 137](#)
- [IEEE 802.1X Device Roles, on page 138](#)
- [Understanding 802.1X Port-Based Authentication, on page 138](#)
- [802.1X host-modes, on page 139](#)
- [Prerequisites for 802.1X Port-Based Authentication, on page 140](#)
- [802.1X with Remote RADIUS Authentication, on page 140](#)
- [802.1X with Local EAP Authentication, on page 142](#)
- [Router as 802.1X Supplicant, on page 145](#)
- [Verify 802.1X Port-Based Authentication, on page 146](#)

Restrictions for IEEE 802.1X Port-Based Authentication

The following restrictions are applicable for IEEE 802.1X port-based authentication:

- 802.1X VLAN assignment is not supported.
- Only single tag dot1q VLAN sub-interfaces are supported.
- Walled-garden VLAN and policies on authentication failures are not supported.

- Subinterfaces and VLAN-tagged traffic are not supported on the ports on which 802.1X port-based authentication is configured. However, this restriction is not applicable from Cisco IOS XR Software Release 7.2.1.
- 802.1X authentication is supported only on physical interfaces.

**Note**

- Communication with the RADIUS server that is initiated by the 802.1x authenticator (RADIUS client) must happen through the built-in management interface on the route processor (RP). Currently, the scenario in which the 802.1x authenticator (RADIUS client) uses a line card port to communicate with the RADIUS server is not supported.

The note is not applicable from Cisco IOS XR Software Release 7.2.1.

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 NCS 540 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.

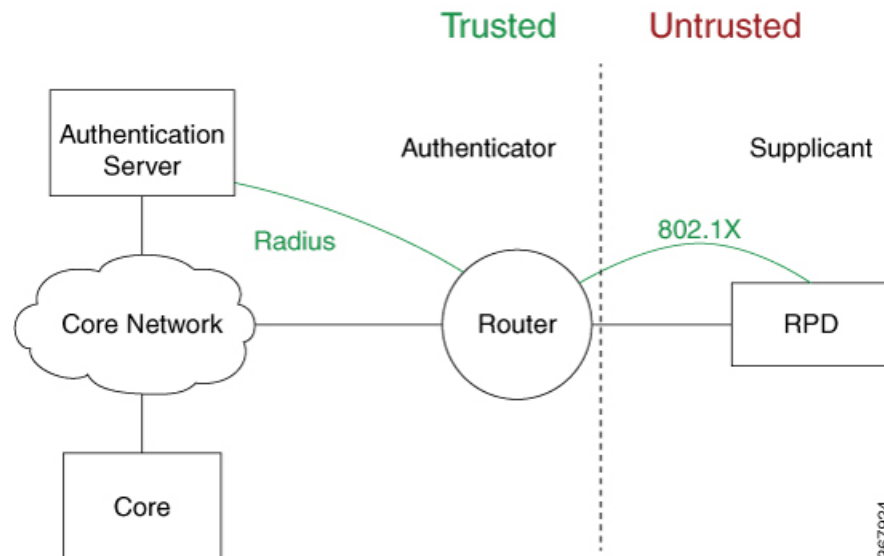
Until the supplicant is authenticated, the port is in *Unauthorized* state, and 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPoL) packets through the port. EAPoL frames can have either default EtherType of 0x888E or Cisco-defined EtherType of 0x876F. After successful 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 3: Topology for IEEE 802.1X Port-Based Authentication



Starting Cisco IOS XR Software Release 7.2.1, 802.1X supports multi-auth and multi-host modes to allow multiple hosts or MAC addresses on a single port. 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. For more information, see [Configure 802.1X host-modes, on page 140](#). 802.1X port-control is also supported on pre-configured VLAN sub-interfaces along with multi-auth and multi-host modes. For VLAN sub-interfaces with VLAN IDs to be pre-configured, VLAN tagged traffic is allowed only after successful 802.1X authentication of the port. There is no default VLAN assignment for untagged traffic.



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 9: 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.

Host modes	Description
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 17](#) and [Configure RADIUS Server Groups, on page 24](#) 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 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 auth
```

Example configuration to allow tagged traffic with VLAN IDs 1 & 2:

```
interface HundredGigE0/3/0/0.1
ip4 address 20.10.1.2 255.255.255.0
encapsulation dot1q 1
!
interface HundredGigE0/3/0/0.2
ip4 address 20.10.2.2 255.255.255.0
encapsulation dot1q 2
!
```

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 67](#) 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 68](#) 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 68](#) 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

```
Router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
    dot1x profile local_auth
```

Example configuration to allow tagged traffic with VLAN IDs 1 & 2:

```
interface HundredGigE0/3/0/0.1
ip4 address 20.10.1.2 255.255.255.0
encapsulation dot1q 1
!
interface HundredGigE0/3/0/0.2
ip4 address 20.10.2.2 255.255.255.0
encapsulation dot1q 2
!
```

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 142](#)
- Trust point: [Configure Trustpoint, on page 143](#)
- Domain name: [Configure Domain Name, on page 143](#)
- Certificates: [Certificate Configurations, on page 143](#)
- EAP profile: [Configure EAP Profile, on page 144](#)

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 7

MACSec Using EAP-TLS Authentication

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
MACsec Using EAP-TLS Authentication	Release 7.3.1	<p>IEEE 802.1X authentication mechanism for Port-Based Network Access Control is the first line of defense against outside attack. It is also one of the most commonly targeted attack vectors, which makes it a security priority.</p> <p>In this release, EAP-TLS, a certificate-based authentication method is used to send client information through IEEE 802.1X, thereby resulting in a secured network connection.</p>



Note MACsec Using EAP-TLS Authentication is supported only on the following variants of Cisco NCS 540 routers:

- N540-24Z8Q2C-SYS
- N540-24Z8Q2C-M

This chapter describes how to achieve MACSec encryption between two Routers using the 802.1X port-based authentication with Extensible Authentication Protocol-Transport Layer Security (EAP-TLS). EAP-TLS allows mutual authentication using certificates, between the authentication server and the client, and generates the Master Session Key (MSK). This MSK is used to derive the Connectivity Association Key (CAK), and the corresponding Connectivity Association Key Name (CKN) is derived from the EAP session ID.

- [Configure MACSec Encryption Using EAP-TLS Authentication, on page 150](#)

Configure MACSec Encryption Using EAP-TLS Authentication

The system supports certificate-based MACsec encryption using both local and remote EAP-TLS authentications.

Restrictions for MACSec Using EAP-TLS Authentication

- The system does not support certificate-based (EAP-TLS) MACsec encryption on sub-interfaces.
- The system does not support MACSec using EAP-TLS authentication in **multi-auth** host mode.

You must also follow the guidelines and restrictions applicable to EAP-TLS session. For details, see the [Restrictions for IEEE 802.1X Port-Based Authentication, on page 137](#) section in the *802.1X Port-Based Authentication* chapter.

Prerequisites

For MACSec using EAP-TLS authentication, you must first configure a EAP-TLS session. For more information on configuring EAP-TLS session, see the following topics in the *802.1X Port-Based Authentication* chapter:

- [802.1X with Remote RADIUS Authentication, on page 140](#)
- [802.1X with Local EAP Authentication, on page 142](#)
- [Router as 802.1X Supplicant, on page 145](#)

The MKA participant with 802.1X PAE role as **authenticator** acts as the key server and the **supplicant** acts as the non-key server.

Configure MACSec EAP on an Interface

The following section describes the steps to configure MACSec EAP on an interface.

Configuration Example

```
Router#configure
Router(config)#interface HundredGigE 0/1/1/2
Router(config-if)#macsec eap
Router(config-if)#commit
```

Running Configuration

```
Router#show run interface HundredGigE 0/1/1/2
interface HundredGigE 0/1/1/2
    macsec eap
!
```

You can also configure MACSec EAP on an interface by specifying the configured MACSec policy name.

Configuration Example

```
Router(config-if)#macsec eap policy test-macsec-policy
```


Running Configuration

```
Router#show run interface HundredGigE 0/1/1/2
interface HundredGigE 0/1/1/2
    macsec eap policy test-macsec-policy
!
```

Verify MACSec EAP Configuration on an Interface

You can use these commands to verify the MACSec EAP configuration:

- **show macsec mka session interface**

Sample output:

```
Router# show macsec mka session interface HundredGigE 0/1/1/2
=====
Interface-Name   Local-TxSCI      #Peers Status  Key-Server PSK/EAP CKN
=====
Hu0/1/12         0201.9ab0.85af/0001  1   Secured YES          EAP      A94399 ...
```

- **show macsec mka session interface detail**

Sample output:

```
Router# show macsec mka session interface HundredGigE 0/1/1/2 detail

MKA Detailed Status for MKA Session
=====
Status : SECURED - Secured MKA Session with MACsec

Local Tx-SCI : 0201.9ab0.85af/0001
Local Tx-SSCI : 2
Interface MAC Address : 0201.9ab0.85af
MKA Port Identifier : 1
Interface Name : Hu0/1/1/2
CAK Name (CKN) : A94399EE68B2A455F85527A4309485DA
CA Authentication Mode : EAP
Keychain : NA (EAP mode)
Member Identifier (MI) : 3222A4A7678A6BDA553FDB54
Message Number (MN) : 114
Authenticator : YES
Key Server : YES
MKA Cipher Suite : AES-128-CMAC
Configured MACSec Cipher Suite : GCM-AES-XPB-256
Latest SAK Status : Rx & Tx
Latest SAK AN : 1
Latest SAK KI (KN) : 3222A4A7678A6BDA553FDB5400000001 (1)
Old SAK Status : No Rx, No Tx
Old SAK AN : 0
Old SAK KI (KN) : RETIRED (0)
SAK Transmit Wait Time : 0s (Not waiting for any peers to respond)
SAK Retire Time : 0s (No Old SAK to retire)
Time to SAK Rekey : NA
MKA Policy Name : *DEFAULT POLICY*
Key Server Priority : 16
Delay Protection : FALSE
Replay Window Size : 64
Include ICV Indicator : FALSE
Confidentiality Offset : 0
Algorithm Agility : 80C201
```

```
SAK Cipher Suite           : 0080C20001000004 (GCM-AES-XPN-256)
MACsec Capability         : 3 (MACsec Integrity, Confidentiality, &
Offset)
MACsec Desired           : YES
```

```
# of MACsec Capable Live Peers      : 1
# of MACsec Capable Live Peers Responded : 1
```

Live Peer List:

MI	MN	Rx-SCI (Peer)	SSCI	KS-Priority
86B47DE76B42D9D7AB6805F7	113	0257.3fae.5cda/0001	1	16

Potential Peer List:

MI	MN	Rx-SCI (Peer)	SSCI	KS-Priority
----	----	---------------	------	-------------

Peers Status:

```
Last Tx MKPDU : 2018 Mar 01 13:36:56.450
Peer Count    : 1
RxSCI         : 02573FAE5CDA0001
MI            : 86B47DE76B42D9D7AB6805F7
Peer CAK      : Match
Latest Rx MKPDU : 2018 Mar 01 13:36:56.450
```

• **show macsec mka summary**

Sample output:

```
Router#show macsec mka summary
=====
Interface-Name Status  Cipher-Suite  KeyChain      PSK/EAP  CKN
=====
Hu0/1/12       Secured      GCM-AES-XPN-256  NA (EAP mode)  EAP      A94399 ...

Total MACSec Sessions : 1
Secured Sessions      : 1
Pending Sessions      : 0
```



CHAPTER 8

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 153](#)

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 line card, as described in this section.



Note The **hw-module fib ipv4 scale internet-optimized** command and **hw-module fib ipv6 scale internet-optimized** command are deprecated from Cisco IOS XR Software Release 7.3.1 and Release 7.4.1, respectively. Hence, if you are upgrading a router (where these configurations are already existing) to Release 7.3.1 or Release 7.4.1 or later, you might see a corresponding warning message stating so.



Note Line cards must be reloaded after disabling the default scale. This is done to ensure that the **hw-module** command configuration takes immediate effect.

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.



CHAPTER 9

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.

- [Implementing Management Plane Protection, on page 155](#)

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

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 0/6/0/10/0/1/0
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 0/6/0/10/0/1/0
```


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 0/6/0/20/0/1/0
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 0/6/0/20/0/1/0
```

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 0/6/0/00/0/1/0
allow all
allow SSH
allow Telnet peer
address ipv4 10.1.0.0/16
!
!
interface HundredGigE 0/6/0/10/0/1/0
allow Telnet peer
address ipv4 10.1.0.0/16
!
!
out-of-band
vrf my_out_of_band
interface HundredGigE 0/6/0/20/0/1/0
allow TFTP peer
address ipv6 33::33
!
!
!
!

show mgmt-plane

Management Plane Protection

inband interfaces
-----

interface - HundredGigE0_6_0_HundredGigE0_0_1_0
ssh configured -
All peers allowed
telnet configured -
peer v4 allowed - 10.1.0.0/16
all configured -
All peers allowed
interface - HundredGigE0_6_0_1_HundredGigE0_0_1_0
telnet configured -

```

```
peer v4 allowed - 10.1.0.0/16

interface - all
  all configured -
    All peers allowed

outband interfaces
-----
interface - HundredGigE0_6_0_2HundredGigE0_0_1_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 10

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.



Note Only the following variants of the Cisco NCS 540 routers support the Traffic Protection for Third-Party Applications:

- N540-ACC-SYS
- N540X-ACC-SYS (Premium)
- N540-24Z8Q2C-M
- N540-28Z4C-SYS

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.

- [gRPC Protocol, on page 165](#)
- [Limitations for Traffic Protection for Third-Party Applications, on page 166](#)
- [Prerequisites for Traffic Protection for Third-Party Applications Over GRPC, on page 166](#)
- [Configuring Traffic Protection for Third-Party Applications, on page 167](#)
- [Troubleshooting Traffic Protection for Third-Party Applications, on page 167](#)

gRPC Protocol

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.

Limitations for Traffic Protection for Third-Party Applications

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 Over 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 verifies whether gRPC TPA is configured or not.

```
Router# show running-config grpc

grpc
no-tls
!
```

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 11

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.



Note Any reference to CiscoSSH in this chapter implies OpenSSH-based implementation of SSH that is available on some of the Cisco IOS XR router variants from Cisco IOS XR Software Release 7.3.2 and later. CiscoSSH replaces Cisco IOS XR SSH, which is the older SSH implementation that existed prior to this release.

For more information, see the *CiscoSSH* section later in this chapter.

- [Implementing Secure Shell, on page 169](#)
- [SSH Port Forwarding, on page 208](#)

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.



Note Any reference to CiscoSSH in this chapter implies OpenSSH-based implementation of SSH that is available on some of the Cisco IOS XR router variants from Cisco IOS XR Software Release 7.3.2 and later. CiscoSSH replaces Cisco IOS XR SSH, which is the older SSH implementation that existed prior to this release.

For more information, see the *CiscoSSH* section later in this chapter.

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
```

To correctly import the public key into the chassis, the public key must be in one of the following formats:

- SSH RSA Public key format
- Base64 encoded (binary) format
- PEM PKCS1 format
- PEM PKCS8 format



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 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 174](#).

- 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.



Note This feature is not supported on the following variants of Cisco NCS 540 Series Routers:

- 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
-

CiscoSSH

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
CiscoSSH	Release 7.3.2	<p>This release introduces CiscoSSH, a newer implementation of SSH on NCS 540 Small Density, NCS 540 Large Density, NCS 540 Medium Density (except N540-ACC-SYS, NC540X-ACC-SYS, and N540-24Z8Q2C-SYS).</p> <p>CiscoSSH leverages OpenSSH implementation, by using the Linux TCP/IP stack to transmit and receive SSH packets over the management Ethernet interface and line card interfaces on the router. CiscoSSH provides additional security features like FIPS compliance and X.509 digital certification. It supports packet path features like MPP, ACL and VRF support, and ensures interoperability with various existing SSH implementations.</p> <p>Note Cisco IOS XR SSH, the SSH implementation that existed prior to this release, is now deprecated on all variants of NCS 540 except for N540-ACC-SYS, NC540X-ACC-SYS, and N540-24Z8Q2C-SYS.</p>



Note CiscoSSH is *not* supported on the following Cisco NCS 540 router variants:

- N540-ACC-SYS
- N540X-ACC-SYS (Premium)
- N540-24Z8Q2C-SYS



Note Any reference to CiscoSSH in this chapter implies OpenSSH-based implementation of SSH that is available on this platform from Cisco IOS XR Software Release 7.3.2 and later. CiscoSSH replaces Cisco IOS XR SSH, which is the older SSH implementation that existed prior to this release.

OpenSSH is a stable, widely deployed open-source implementation of SSH. CiscoSSH implementation leverages the key features of openSSH such as strong authentication, cryptography, encryption, port forwarding, and so on, to provide secured management access to the router. CiscoSSH provides additional security features like FIPS compliance and support for X.509 digital certificate.

For more information on SSH in general, see [Information About Implementing Secure Shell, on page 170](#) and [Configure SSH, on page 183](#).

The CiscoSSH implementation also ensures backward compatibility for all the existing Cisco IOS XR SSH configuration and management. You can continue to use SSH the way it was existing before. The functionality and configuration commands of CiscoSSH and Cisco IOS XR SSH remain the same for majority of the part. However, certain behavioral changes exist between CiscoSSH and Cisco IOS XR SSH. For details, see the subsequent sections.

This table lists the behavioral changes introduced by CiscoSSH as compared to Cisco IOS XR SSH. Also, see [Guidelines for Using CiscoSSH, on page 179](#).

Table 12: Behavioral Changes Introduced by CiscoSSH in Comparison to Cisco IOS XR SSH

Functionality	CiscoSSH	Cisco IOS XR SSH
Port number for Netconf server	The system uses the port numbers 830 (the default IANA-assigned TCP port number for Netconf over SSH) or 22 (the default port number for SSH) for the Netconf server. You cannot configure this value.	You can explicitly configure the desired port number for Netconf server using the ssh server netconf port command.
Username syntax	Because CiscoSSH considers ':' (<i>colon</i>) as a delimiter in certain types of user authentication, it does not support authentication of usernames having ':' (<i>colon</i>) in it.	No restriction for using ':' (<i>colon</i>) in username syntax.
Configuring unsupported algorithms	You cannot enable unsupported algorithms using any configuration command.	You can explicitly enable unsupported algorithms using the ssh server enable cipher command.
SSH session timeout	The SSH session initiated from the router to an unreachable host times out after 120 seconds.	The SSH session initiated from the router to an unreachable host times out after 60 seconds.
SSH session timeout criteria	The SSH timeout configuration considers the total timeout value for the maximum number of login attempts allowed.	The SSH timeout configuration considers the timeout value for individual login attempt.

Functionality	CiscoSSH	Cisco IOS XR SSH
Time-based rekey of SSH sessions	The router triggers time-based rekey of SSH sessions only when it receives a packet after the timer expiry.	The router triggers time-based rekey of SSH sessions immediately after the timer expiry.
LPTS policer rate for port-forwarded SSH sessions	When using SSH port forwarding feature, the router considers the traffic flows corresponding to port-forwarded SSH sessions as third party applications. Hence, the LPTS polices those traffic flows at a medium rate.	The LPTS polices the traffic flows corresponding to port-forwarded SSH sessions at a high rate.
Port-forwarded channels	No limit to the number of port-forwarded channels supported with CiscoSSH. But, the show ssh command displays a maximum of only 16 entries.	Supports a maximum of 16 port-forwarded channels.
File transfer through SCP	While using SCP with CiscoSSH, the router checks for the presence of system files after authentication.	The router checks for the presence of system files before authentication.
File transfer through SFTP	With non-interactive SFTP session initiated from the router, you can transfer files from an external device to the router; not from the router to external device.	You can transfer files from an external device to the router, and the other way round.

Restrictions for Cisco SSH

- Does not support SSH version 1
- Does not support back up SSH server
- Does not allow to use secondary IPv4 addresses because they are not currently synchronized to Linux
- Does not support BVI interfaces as source or destination for the SSH connections
- Does not support these algorithms:
 - The cipher algorithms, *aes128-cbc*, *aes192-cbc*, *aes256-cbc*, and *3des-cbc*
 - The key-exchange algorithm, *diffie-hellman-group1-sha1*
- Does not support these commands:
 - **show ssh history**
 - **show ssh history details**
 - **clear ssh stale sessions**

Guidelines for Using CiscoSSH

The following section lists certain functionality aspects and guidelines for using CiscoSSH.

- **Netconf Request:** You must follow a specific syntax when you send Netconf request over CLI. Add the subsystem (*netconf* or *sftp*) name as the last argument while issuing an SSH command.

For example,

```
ssh username@ipaddress -p 830 -s netconf ---> Correct usage
ssh username@ipaddress netconf -p 830 -s ---> Incorrect usage
```

- **Configuring unsupported algorithms:** Configuring CiscoSSH server only with unsupported algorithms (*3des-cbc* or *diffie-hellman-group1-sha1*) results in commit failure. Hence, you must remove such configurations on your router as a part of the pre-upgrade procedure.

For example,

```
Router(config)#ssh server algorithms cipher 3des-cbc

!!% Operation not permitted: 3des-cbc is not supported in ciscossh, SSH cannot work
with this option only
```

Similarly, if you configure CiscoSSH server with both supported and unsupported algorithms, then the router issues the following warning and removes the unsupported algorithm:

```
Router(config)#ssh server algorithms cipher aes128-ctr aes192-ctr 3des-cbc

ssh_conf_proxy[1193]: %SECURITY-SSHD_CONF_PRX-3-ERR_GENERAL : 3des-cbc is not supported,
will be removed
```

- **SSH session keep alive:** By default, the SSH session keep alive functionality is enabled in CiscoSSH, to detect and terminate unresponsive sessions. The default keep alive time is 60 seconds, with a maximum of three attempts allowed, so that the detection time for unresponsive sessions is 180 seconds. These keep alive parameters are not configurable.
- **TCP window scale:** Although the router accepts the configuration to change the TCP window scale parameter, the configuration does not have any effect with CiscoSSH. This is because, CiscoSSH uses Linux TCP/IP stack that has dynamic window scaling, and hence it does not require applications to specify the window scale.
- **SSH session limit and rate limit:** Although the configuration for SSH session limit and rate limit applies to all VRFs where SSH is enabled, the router enforces the limit for each VRF. However, the maximum number of virtual teletype (VTY) sessions across all VRFs still remains as 200. This in turn limits the total number of SSH sessions that require a VTY interface, across all VRFs. As a result, when upgrading from a release version having Cisco IOS XR SSH to a version having CiscoSSH, the system applies the session limit and rate limit configurations to all VRFs where SSH is enabled. Hence, as part of the post-upgrade procedure, you must reconfigure these limits to achieve the same limit as that of Cisco IOS XR SSH.
- **SSH session limit enforcement:** Information on the number of active SSH sessions on the router is not persistent across SSH server process restarts. Hence, SSH session limit enforcement does not consider the existing sessions after an SSH server restart.
- **SSH with ACL or MPP configuration:** With SSH ACL or MPP configured on the router, the attempt for client connection that is now allowed as per that configuration times out. The router does not send TCP reset for such blocked SSH connections. This implementation is to enhance security.

- **Default VRFs:** Configuring the default SSH VRF using the `ssh vrf default` command enables only version 2 of CiscoSSH, because version 1 is not supported.
- **Non-default VRFs:** If SSH service is enabled on any of the non-default VRFs that is configured on the router, and if you restart the `ssh_conf_proxy` process, there might be a delay in allowing incoming SSH sessions on that non-default VRF. The session establishment might even timeout in such a scenario. This behavior is due to the delay in programming the LPTS entries for those sessions.
- **Public key-based authentication:** In CiscoSSH, the router negotiates public key-based authentication even if there is no public key imported on to the router. So, the authentication attempt from the client using public key fails in such scenarios. The router displays a syslog on the console for this authentication failure. However, the client and server proceed with subsequent authentication methods like keyboard-interactive and password methods. If the router does not have a public key imported, you may choose to disable public key-based authentication from the client side. For details on public key-based authentication, see the *Implementing Certification Authority Interoperability* chapter in this guide.
- **Modifying SSH configuration:** Any change to the SSH configuration results in process restart of SSH server process. However, it does not impact the existing SSH, SCP, SFTP, or Netconf sessions.
- **Clearing SSH sessions:** The `clear ssh all` command clears all incoming sessions.
- **Line-feed option:** Adding a line-feed option for Gossh-based clients results in SSH session establishment failure. This is because, the SSH client checks for non-zero window size for session establishment. Whereas CiscoSSH sends window size as 0. The workaround for this issue is to use the option to ignore the window size while initiating an SSH connection from such clients.
- **Virtual IP addresses:** After a process restart of `xlncd` or `ip_smiap`, there might be a delay in restoring the virtual IP addresses.
- **More-specific Routes:** Routes that are more specific than a connected route will not be available through Linux.

For example:

XR routing table:

```
10.0.0.0/24 via 10.0.0.2 (connected route)
10.0.0.192/28 via 20.0.0.1 (static route)
```

The expected behavior is as follows:

Table 13: Expected Behavior of More-specific Routes with CiscoSSH

Destination IP Range	Cisco IOS XR OS Sends to:	Linux Sends to:	Match (Yes/No)
10.0.0.1 - 10.0.0.191	10.0.0.2	10.0.0.2	Yes
10.0.0.193 - 10.0.0.206	20.0.0.1	10.0.0.2	No
10.0.0.207 - 10.0.0.255	10.0.0.2	10.0.0.2	Yes

- **Verification commands:** During stress test on the router, certain show commands like `show ssh`, `show ssh session details`, and `show ssh rekey` might time out. The console displays the following error message in such cases:

"Error: Timed out to obtain information about one or more incoming/outgoing session. please retry."

• **Process restart:**

- You cannot restart the CiscoSSH server process using the **process restart ssh_server** command, because it is a Linux process. Use the **kill** command on the Linux shell to restart the process.
- CiscoSSH has *ssh_conf_proxy* and *ssh_syslog_proxy* processes that are responsible for processing the SSH configuration and logging syslog messages respectively. You can restart these processes using the **process restart** command.
- A restart of *XR-TCP* process does not have any impact on CiscoSSH functionality, because CiscoSSH uses Linux TCP.

• **Debuggability:**

- You can enable 3 levels of debugs for CiscoSSH using the **debug ssh server 1/12/13** command. Similarly, you can use the **debug ssh client 1/12/13** command for CiscoSSH client.
- The SSH server process restarts every time you enable or disable the debugs, because enabling the debugs results in updating the LOGLEVEL in the internal *sshd_config* file.

Syslogs for CiscoSSH

CiscoSSH introduces new syslogs for various SSH session events. The following table gives a comparison of syslogs between CiscoSSH and Cisco IOS XR SSH:

Table 14: Syslogs for CiscoSSH and Cisco IOS XR SSH

Session Event	Syslogs on CiscoSSH	Syslogs on Cisco IOS XR SSH
Session login	<pre>RP/0/RP0/CPU0:Sep 22 11:06:33.467 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[32504]: Accepted authentication/pam for admin from 203.0.113.1 port 62015 ssh2 RP/0/RP0/CPU0:Sep 22 11:06:33.472 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[32504]: User child is on pid 32564 RP/0/RP0/CPU0:Sep 22 11:06:33.519 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[32564]: Starting session: shell on pts/1 for admin from 203.0.113.1 port 62015 id 0</pre>	<pre>RP/0/RP0/CPU0:Sep 22 11:46:13.475 IST: SSHD_[67274]: %SECURITY-SSHD-6-INFO_SUCCESS : Successfully authenticated user 'root' from '192.0.2.1' on 'vty0'(cipher 'aes128-ctr', mac 'hmac-sha2-256')</pre>

Session Event	Syslogs on CiscoSSH	Syslogs on Cisco IOS XR SSH
Session logout	<pre>RP/0/RP0/CPU0:Sep 22 11:11:27.394 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[32564]: Received disconnect from 203.0.113.1 port 62015:11: disconnected by user RP/0/RP0/CPU0:Sep 22 11:11:27.394 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[32564]: Disconnected from user admin 203.0.113.1 port 62015</pre>	<pre>RP/0/RP0/CPU0:Sep 22 11:46:48.439 IST: SSHD_[67274]: %SECURITY-SSHD-6-INFO_USER_LOGOUT : User 'root' from '192.0.2.1' logged out on 'vty0'</pre>
Session login failure	<pre>RP/0/RP0/CPU0:Sep 22 19:47:06.211 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[31103]: Failed authentication/pam for admin from 203.0.113.1 port 60189 ssh2</pre>	<pre>RP/0/RP0/CPU0:Sep 22 11:47:55.909 IST: SSHD_[67369]: %SECURITY-SSHD-4-INFO_FAILURE : Failed authentication attempt by user 'root' from '192.0.2.1' on 'vty0'</pre>
Session rekey	<pre>ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PFX-6-INFO_GENERAL : sshd[24919]: Server initiated time rekey for session=21, session_rekey_count =1</pre>	<pre>RP/0/RP0/CPU0:Sep 22 19:07:45.435 IST: SSHD_[65640]: %SECURITY-SSHD-6-INFO_REKEY : Server initiated time rekey for session 4 , session_rekey_count = 1</pre>

Restrictions for Implementing Secure Shell

The following are some basic SSH 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.
- 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 1 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 explicitly. See, [Automatic Generation of SSH Host-Key Pairs, on page 187](#) 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
-

Automatic Generation of SSH Host-Key Pairs

This feature brings in the functionality of automatically generating the SSH host-key pairs for the DSA, ECDSA (such as **ecdsa-nistp256**, **ecdsa-nistp384**, and **ecdsa-nistp521**) and RSA algorithms. This in turn eliminates the need for explicitly generating each SSH host-key pair after the router boots up. Because the keys are already present in the system, the SSH client can establish connection with the SSH server soon after the router boots up with the basic SSH configuration. This is useful especially during zero touch provisioning (ZTP) and Golden ISO boot up scenarios.

Before this automation, you had to execute the **crypto key generate** command to generate the required host-key pairs.

Although the host-key pairs are auto-generated with the introduction of this feature, you still have the flexibility to select only the required algorithms on the SSH server. You can use the **ssh server algorithms host-key** command in XR Config mode to achieve the same. Alternatively, you can also use the existing **crypto key zeroize** command in XR EXEC mode to remove the algorithms that are not required.

Prior to the introduction of this feature, you had to execute the **crypto key generate** command in XR EXEC mode to generate the required host-key pairs.



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.



Note This feature is not supported on the following variants of Cisco NCS 540 Series Routers:

- 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
-

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

[Automatic Generation of SSH Host-Key Pairs, on page 187](#)

Associated Commands

- `ssh server algorithms host-key`
- `show crypto key mypubkey`

Ed25519 Public-Key Signature Algorithm Support for SSH

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Ed25519 Public-Key Signature Algorithm Support for SSH	Release 7.3.1	<p>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.</p> <p>This command is modified for this feature:</p> <p>ssh server algorithms host-key</p>

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
```

Configuring CBC Mode Ciphers

In Cisco IOS XR Release 7.0.1, you can enable CBC mode ciphers 3DES-CBC and AES-CBC for SSHv2 server and client connections. The ciphers are disabled by default.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
Enters mode.
```

Step 2 **ssh server enable cipher aes-cbc 3des-cbc**

Example:

```
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
```

Step 3 **ssh client enable cipher aes-cbc 3des-cbc****Example:**

```
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
```

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 **show ssh session details****Example:**

```
Router# show ssh session details
```

Configuring CBC Mode Ciphers

```
/*Enable CBC mode ciphers 3DES-CBC and AES-CBC */
Router# configure
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
Router(config)# commit
```

Verify CBC Mode Cipher Configuration.

```
Router# show ssh session details
```

```
Thu Sep 6 10:16:26.346 UTC
SSH version : Cisco-2.0
```

id	key-exchange	pubkey	incipher	outcipher	inmac	outmac

Incoming Session						
2	ecdh-sha2-nistp256	ssh-rsa	aes128-cbc	aes128-cbc	hmac-sha2-256	hmac-sha2-256

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.



Note This feature is not supported on the following variants of Cisco NCS 540 Series Routers:

- 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
-

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-shal
Router(config)#commit
```

```
Router(config)# ssh client disable hmac hmac-shal
Router(config)#commit
```

Running Configuration

```
ssh server disable hmac hmac-shal
!
```

```
ssh client disable hmac hmac-shal
!
```

Related Topics

[SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 193](#)

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 193](#)

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 16: Feature History Table

Feature Name	Release Information	Feature Description
User Configurable Maximum Authentication Attempts for SSH	Release 7.3.1	<p>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.</p> <p>This command is introduced for this feature:</p> <p>ssh server max-auth-limit</p>

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 17: Feature History Table

Feature Name	Release Information	Feature Description
X.509v3 Certificate-based Authentication for SSH	Release 7.3.1	<p>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.</p> <p>Commands introduced for this feature are:</p> <p>ssh server certificate</p> <p>ssh server trustpoint</p> <p>This command is modified for this feature:</p> <p>ssh server algorithms host-key</p>

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.

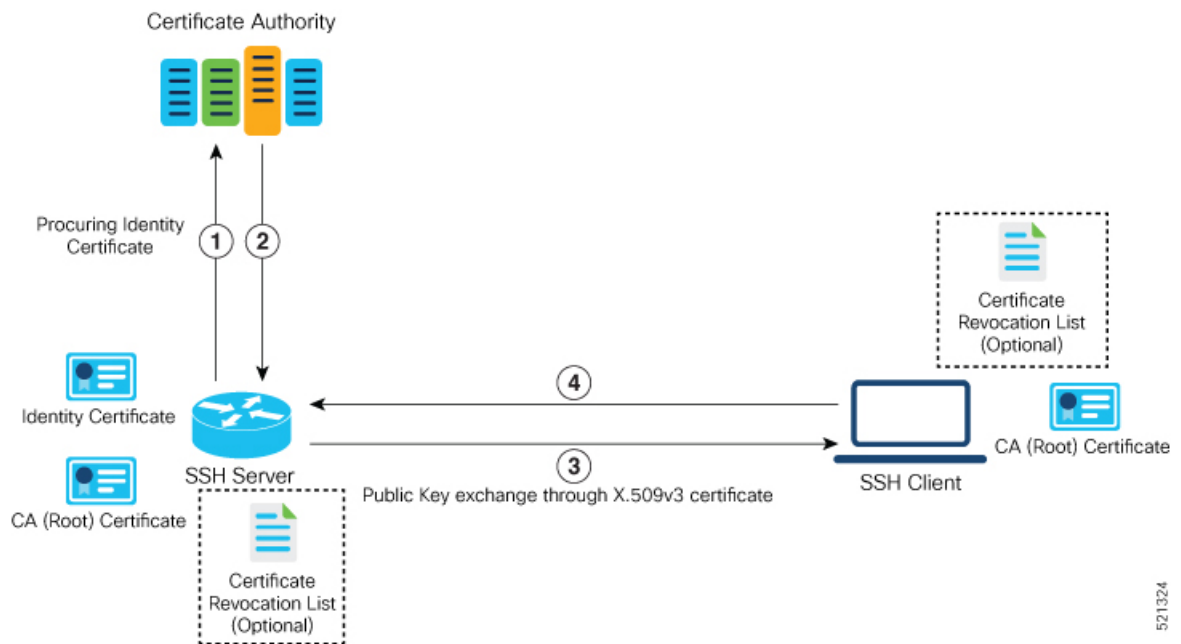
There are two SSH protocols that use public-key cryptography for authentication:

- 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.

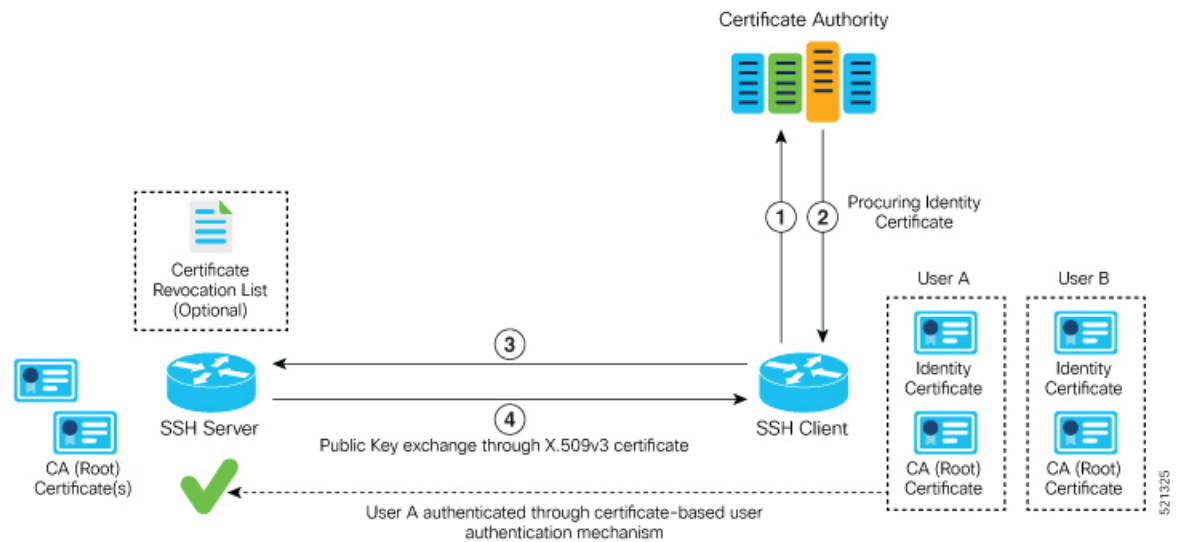
Server Authentication using X.509v3 Certificate



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(othertype)** 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 lists 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 verification 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 199](#)

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`

Importing Public Key

Public key can be imported in EXEC or any configuration modes. To correctly import a public key into the chassis, the public key must be in one of the following formats:

- SSH RSA Public key format—Supported in configuration mode and in EXEC mode as text and file.
- Base64 encoded (binary) format—Supported as EXEC mode as file.



Note Base64 format is not supported if you press enter and copy paste the key.

- PEM PKCS8 format—Supported in EXEC mode as text and file.
- PEM PKCS1 format—Supported in EXEC mode as text and file.

Only the users belonging to user groups with admin and crypto task ID can import public key for other users.

Only the users belonging to groups root-lr, netadmin, maintenance, sysadmin can import public key for other users.

Import Public Key in EXEC Mode

Use the following command to import public key for currently logged in user:

```
Crypto key import authentication rsa file_path
Crypto key import authentication rsa <press enter here>
```

Use the following command to import public key for other users:


```
Crypto key import authentication rsa username username file_path
Crypto key authentication rsa username <press enter here>
```

Remove Public Key Imported in EXEC Mode

Public Keys imported in EXEC mode can be zeroized and removed by using the following commands:

- For currently logged in user:

```
crypto key zeroize authentication rsa
```

- For other users:

```
crypto key zeroize authentication rsa username username
```

- For all users:

```
crypto key zeroize authentication rsa all
```

Import Public Key in Configuration Mode



Note User must be present in the database to which the keys are imported.



Note Only SSH RSA Public key format is supported in configuration mode.

Use the following command to import public key in configuration mode:

```
ssh server username username keystring ssh-rsa pubkey
```

Remove Public Key Imported in Configuration Mode

Use the following command to remove public key in configuration mode:

```
no ssh server username username
```

Verify the Import of Public Key

Use the following commands to display public keys imported from both configuration and EXEC modes.

- For currently logged in user:

```
show crypto key authentication rsa
```

- For other users:

```
show crypto key authentication rsa username username
```

- For all users:

```
show crypto key authentication rsa all
```

SSH Port Forwarding

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
SSH Port Forwarding	Release 7.3.2	<p>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.</p> <p>This feature introduces the ssh server port-forwarding local command.</p> <p>Note For the following Cisco NCS 540 router variants, this feature is supported with Cisco IOS XR SSH:</p> <ul style="list-style-type: none"> • N540-ACC-SYS • N540X-ACC-SYS (Premium) • N540-24Z8Q2C-SYS <p>For all other Cisco NCS 540 router variants, this feature is supported with CiscoSSH, an OpenSSH-based implementation of SSH.</p>

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 4: SSH Port Forwarding

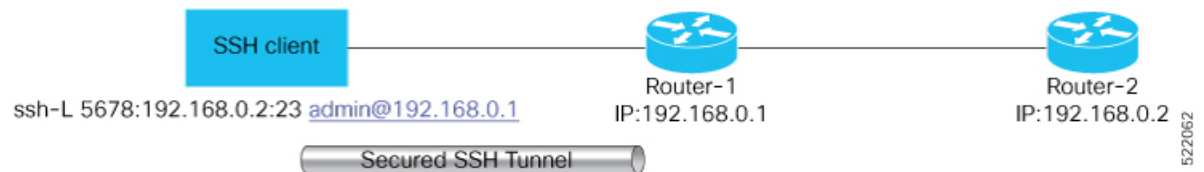


Consider an application on the SSH client residing on a local host, trying to connect to an application server residing on a remote host. With tunneling enabled, the application on the SSH client connects to a port on the local host that the SSH client listens to. The SSH client then forwards the data 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 5: 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 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**



CHAPTER 12

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 **ncs540-li-1.0.0.0-r632361.x86_64.rpm**.

For the following Cisco NCS 540 router variants, the Lawful Intercept package is present in the base package; no separate RPMs are required:

- 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

For more information about activating and deactivating the Lawful Intercept package, see the [Installing Lawful Intercept \(LI\) Package, on page 217](#) section.

- [Information About Lawful Intercept Implementation, on page 214](#)
- [Prerequisites for Implementing Lawful Intercept, on page 214](#)
- [Restrictions for Implementing Lawful Intercept, on page 215](#)
- [Lawful Intercept Topology, on page 216](#)
- [Benefits of Lawful Intercept, on page 216](#)
- [Installing Lawful Intercept \(LI\) Package, on page 217](#)
- [How to Configure SNMPv3 Access for Lawful Intercept, on page 218](#)
- [Additional Information on Lawful Intercept, on page 220](#)

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.
- **QoS Peering**— QoS peering must be enabled on the router for Lawful Intercept to work.



Note The Lawful Intercept feature has no intersection with the QoS feature on the router. Enabling the QoS peering profile with **hw-module profile qos ingress-model peering** command on all the required line cards, allows QoS and Lawful intercept to allocate hardware resources.

- 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.

Restrictions for Implementing Lawful Intercept

The following restrictions are applicable for Lawful Intercept:

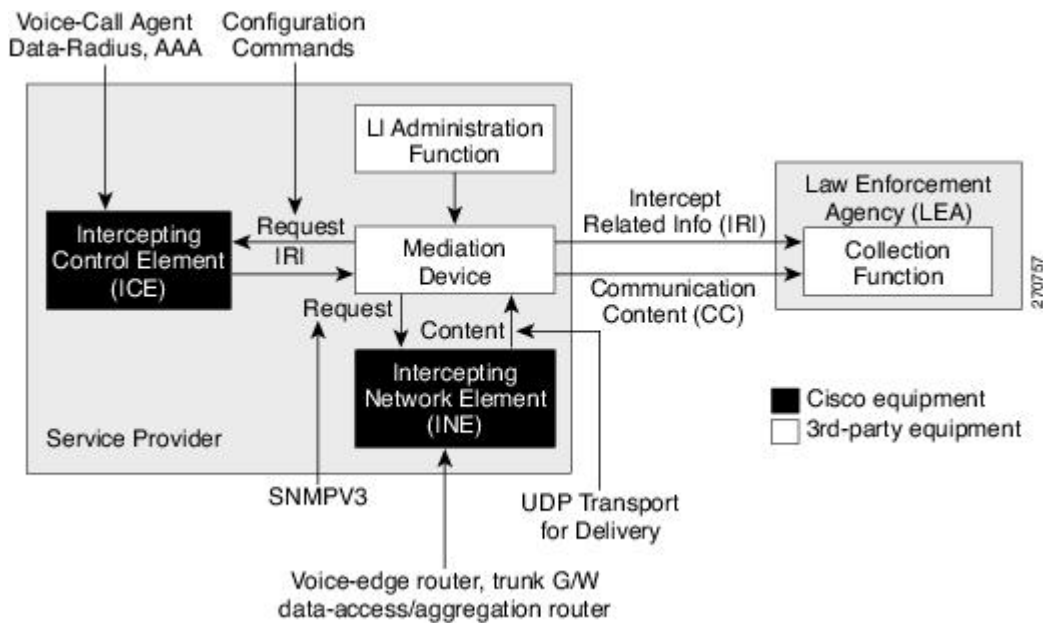
- Lawful Intercept shares a pool of 16 unique source IP addresses with tunnel-ip. The combined configuration of GRE tunnel-ips and the MDs (the cTap2MediationSrcInterface field) shall not yield more than 16 unique source IPs. Note that when configuring the MD, if the value 0 is passed in for the cTap2MediationSrcInterface 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 pure IP over Ethernet packets.
- Only 250 MDs and 500 Taps of IPv4 and IPv6 each are supported.
- One Tap-to-multiple MDs is not supported.
- After the route processor reload or fail-over, the MD and Tap configuration must be re-provisioned.
- Only IPv4 MD is supported.
- 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.
- In Cisco IOS XR Release 6.3.x, QoS peering must be enabled for QoS to work.
In Cisco IOS XR Release 6.5.x and later, QoS peering is not required.
- 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

Lawful Intercept Topology

This figure shows intercept access points and interfaces in a lawful intercept topology for both voice and data interception.

Figure 6: 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.

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 **ncs540-li-1.0.0.0-r63236I.x86_64.rpm**.

Configuration

```
Router# install add source
tftp://223.255.254.252/auto/tftp-sjc-users/username/ncs540-li-1.0.0.0-r63236I.x86_64.rpmncs560-li-1.0.0.0-r66136I.x86_64.rpm
Router# install activate ncs540-li-1.0.0.0-r63236I.x86_64.rpm
Router# install commit
```

Verification

```
Router# show install active
Node 0/RP0/CPU0 [RP]
  Boot Partition: xr_lv0
  Active Packages: 2
    ncs540-xr-6.3.2.36I version=6.3.2.36I [Boot image]
    ncs540-li-1.0.0.0-r63236I.x86_64.rpm

Node 0/0/CPU0 [LC]
  Boot Partition: xr_lcp_lv0
  Active Packages: 2
    ncs540-xr-6.3.2.36I version=6.3.2.36I [Boot image]
    ncs540-li-1.0.0.0-r63236I.x86_64.rpm
```

Deactivating the LI RPM



Note You might experience interface or protocol flaps while uninstalling or deactivating the LI RPM. Hence, we recommend you to perform this activity during a maintenance window.

To uninstall the Lawful Intercept package, deactivate `ncs540-li-1.0.0.0-r63236I.x86_64.rpm` as shown in the following steps:

Configuration

```
Router# install deactivate ncs540-li-1.0.0.0-r63236I.x86_64.rpm
Router# install commit
Router# install remove ncs540-li-1.0.0.0-r63236I.x86_64.rpm
Router# show install committed
```

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 `ncs540-li-1.0.0.0-r63236I.x86_64.rpm`.



Note For the following Cisco NCS 540 router variants, the Lawful Intercept package is present in the base package; no separate RPMs are required:

- 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

- 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 For the following Cisco NCS 540 router variants, the Lawful Intercept package is present in the base package; no separate RPMs are required:

- 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

The **lawful-intercept disable** command is available on the router, only after installing and activating the **ncs540-li-1.0.0.0-r63236L.x86_64.rpm**.

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

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.

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.

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 interface 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 cTapStreamEntry, 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 13

Need for Trustworthy Systems

In Cisco IOS XR Release 7.0.1, this section is applicable *only* to the following Cisco NCS 540 variants:

- N540-28Z4C-SYS-A/D
- N540X-16Z4G8Q2C-A/D
- N540-12Z20G-SYS-A/D
- N540X-12Z16G-SYS-A/D

Starting Cisco IOS XR Release 7.3.1, this section is also applicable to the following Cisco NCS 540 variants:

- N540X-6Z18G-SYS-A/D
- N540X-8Z16G-SYS-A/D
- N540-FH-CSR-SYS

Global service providers, enterprises, and government networks rely on the unimpeded operation of complex computing and communications networks. The integrity of the data and IT infrastructure is foundational to maintaining the security of these networks and user trust. With the evolution to anywhere, anytime access to personal data, users expect the same level of access and security on every network. The threat landscape is also changing, with adversaries becoming more aggressive. Protecting networks from attacks by malevolent actors and from counterfeit and tampered products becomes even more crucial.

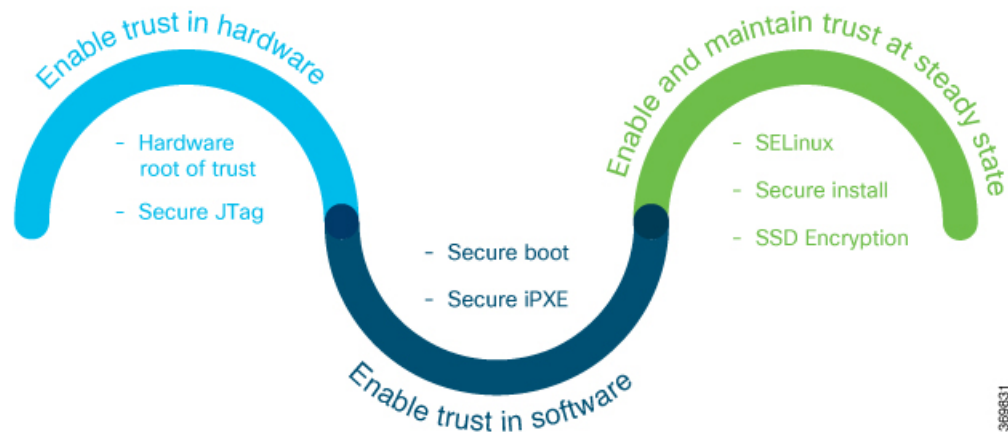
Routers are the critical components of the network infrastructure and must be able to protect the network and report on system integrity. A “trustworthy solution” is one that does what it is *expected* to do in a *verifiable* way. Building trustworthy solutions requires that security is a primary design consideration. Routers that constitute trustworthy systems are a function of security, and trust is about preventing as well as knowing whether systems have been tampered with.

In trustworthy systems, trust starts at the lowest levels of hardware and is carried through the boot process, into the operating system (OS) kernel, and finally into runtime in the OS.

The main components of implementing a trustworthy system are:

- Enabling trust in hardware with Hardware root-of-trust and secure JTAG
- Enabling trust in software with secure boot and secure iPXE
- Enabling and maintaining trust at steady state with Security-Enhanced Linux (SELinux), Secure install, and SSD Encryption

Figure 7: Ecosystem of Trustworthy Systems



Trustworthy systems must have methods to securely measure hardware, firmware, and software components and to securely attest to these secure measurements.

For information on key concepts used in this chapter, see the [Understanding Key Concepts in Security](#).

- [Enable Trust in Hardware, on page 226](#)
- [Enable Trust in Software, on page 230](#)
- [Establish and Maintain Trust at Steady State, on page 231](#)
- [How Trustworthiness is Implemented, on page 247](#)
- [Understanding Key Concepts in Security, on page 248](#)

Enable Trust in Hardware

The first component in implementing a trustworthy system is to enable trust in hardware.

Because software alone can't prove a system's integrity, truly establishing trust must also be done in the hardware using a hardware-anchored root of trust. Without a hardware root of trust, no amount of software signatures or secure software development can protect the underlying system from becoming compromised. To be effective, this root of trust must be based on an immutable hardware component that establishes a chain of trust at boot-time. Each piece of code in the boot process measures and checks the signature of the next stage of the boot process before the software boots.

A hardware-anchored root of trust is achieved through:

- **Anti-counterfeit chip:** All modules that include a CPU, as well as the chassis, are fitted with an anti-counterfeit chip, which supports co-signed secure boot, secure storage, and boot-integrity-visibility. The chip ensures that the device's software and hardware are authentic and haven't been tampered with or modified in any way. It also helps to prevent unauthorized access to the device's sensitive data by enforcing strong authentication and access control policies.
- **Secure Unique Device Identifier (SUDI):** The X.509 SUDI certificate installed at manufacturing provides a unique device identifier. SUDI helps to enable anti-counterfeit checks along with authentication and remote provisioning. The SUDI is generated using a combination of the device's unique hardware identifier (such as its serial number or MAC address) and a private key that is securely stored within the device. This ensures that each SUDI is unique and cannot be easily duplicated or forged. When a device attempts to connect to a network, the network uses the SUDI to authenticate the device, and ensure that it's

authorized to connect. This helps to prevent unauthorized access to the network and ensures that only trusted devices are allowed to connect.

- **Secure JTAG:** The secure JTAG interface is used for debugging and downloading firmware. This interface with asymmetric-key based authentication and verification protocols prevents attackers from modifying firmware or stealing confidential information. Secure JTAG typically involves a combination of hardware and software-based security measures. For example, it may include the use of encryption and authentication protocols to secure communications between the JTAG interface and the debugging tool. It may also involve the use of access control policies and permissions to restrict access to the JTAG interface to authorized users only.



Note Hardware-anchored root of trust is enabled by default on Cisco NCS 540 Series routers.

Verification

You can verify if trust is enabled in the hardware by executing the following command:

```
Router#show platform security integrity hardware
Wed Apr 17 11:19:03.202 UTC

+-----+
Node location: node0_RP0_CPU0
+-----+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 52050
Known-good-digests:
Index  value
  0     hh4jzFB1xSGHZ4hKqnC2FEjqHg4tpx/chZ7YcTwLCco=
observed-digests:
Index  value
  0     hh4jzFB1xSGHZ4hKqnC2FEjqHg4tpx/chZ7YcTwLCco=
PCRs:
Index  value
  15    D11BGskyzeJ1LNYKuZK8Qq1lwkth0ru+0xWydL9YMdc=
```

Secure Hardware for Strong Cryptography

All Cisco IOS XR7 supported-platforms ships with a non-tamperable Trust Anchor module (TAm) in the hardware.

TAm houses known-good-values (KGVs) of the hardware components along with keys and certificates rooted to Cisco, which are used to verify components of the hardware during the BIOS boot.

Chip Guard and Attestation are security features implemented in TAm to enable trust in hardware.

- Chip Guard detects tampering attempts and responds by initiating actions such as disabling access to the device, erasing sensitive information stored in the device, or triggering a security alarm.
- Attestation provides a mechanism for verifying the integrity and authenticity of the software and hardware components of a device.

A Cisco router with SUDI is authenticated and verified remotely for uniquely identifying that it's an authentic Cisco device.



Note Some Cisco NCS 540 Series Routers have the older generation of chips with lesser capabilities compared to the latest TAM chips present on the newer generation of hardware.

Chip Guard for Hardware Integrity

The chip guard feature helps detect if attackers have replaced a Cisco router's ASIC or CPU with a counterfeit ASIC or CPU in the supply chain.

Need for Additional Hardware Integrity Check

The Cisco TAM chip and secure boot process anchored inside the hardware chip are enough to manage any threats around the boot process. Other threat factors include increased supply chain attacks, where attackers replace the ASIC or CPU on a router with malware-infested chips. Once the ASIC or CPU is replaced, the integrity of the hardware is compromised. The chip guard feature helps detect if there is a counterfeit ASIC or CPU on a router.

Stages in the Chip Guard Hardware Integrity Check

The table shows the various stages where the Chip Guard feature is triggered to ensure the integrity of the hardware:

Stage	Process/Action	Result
Chip Manufacturing	SHA 256 hashes of the electronic chip IDs of both the CPU and ASIC are programmed in the TAM chip	The imprint DB inside the TAM chip contains the SHA 256 hashes, which cannot be modified during the router's lifetime.
Router Deployed in the Field and When BIOS is Triggered	The chip guard feature recomputes the SHA 256 hashes of the electronic chip IDs of both the CPU and ASIC and creates an observed DB.	The observed values are stored in the Observed DB inside the TAM chip.

Stage	Process/Action	Result
Comparison of Imprint DB and Observed DB	DBs match	The router continues to boot. Depending on the capability of the underlying router, the chip guard feature validates either the CPU, ASIC, or both.
	DBs do not match	The router indicates that either the CPU or ASIC is modified, and the secure boot process halts. A message is displayed on the console about the chip guard validation failure. Note Visit the Products Returns & Replacement (RMA) website to create a service request.

Attestation

Attestation enables external verifiers to check the integrity of the software running on the host. Implementing this feature on Cisco hardware helps you validate the trustworthiness of the hardware and software of network devices.

Once a router is up and running, you can send a request to an external verifier. The external verifier requests an attestation quote from the router. The TAM chip can output the PCR quote and audit log, and it signs the quote using an attestation private key for that specific router and responds to the verifier. The verifier uses Cisco-provided KGV hashes and the Attestation Public Certificate to verify the attested PCR quotes and audit logs. This verification is protected against repeat attacks using a nonce. Besides this, the verification ensures that the attestation is specific to a particular router by using attestation key pairs. These attestation key pairs are unique to each router. This ensures that attackers do not tamper with the router hardware, boot keys, boot configuration, and running software.

Proof of hardware integrity is recorded in the TAM as part of Chip Guard. This proof is made available through the following command:



Note The same data is also available through NETCONF for a remote attestation server:
Cisco-IOS-XR-remote-attestation-act.yang.

```
RP/0/RP0/CPU0:NCS-540-C-LNT#show platform security attest pcr 0 trustpoint ciscoaik nonce
4567 location 0/RP0/CPU0
Thu Apr 11 05:44:10.808 UTC
Nonce: 4567

+-----+
Node location: node0_RP0_CPU0
+-----+
Uptime: 1198700
pcr-quote:
/1R0FAACyASBFFG5Nuf7DQRMET669lhcdFZWLp0FQCRVAAWBSLQPA97////AQWACQAWLAWAAQIAEWAAgrE798LlK4p1kyA50kG0V46LlQuS6GUjG8yF=
```

```

pcr-quote-signature:
mC8oFWz9STge31DLXCs/Ez7BRKsZyDv04auhJagWHa3aHsa9eVf34Y/EMfIttjeAhs<truncated> dYpsFKMGkrcrolTqufThaDlGfIH+Gh4QBewdNky3Igiw==
pcr-index          pcr-value
0                  sL3H+Em2xzxXrNUoDF+kC47IXxN4V/d/7hYUXApXNoY=
    
```

See the [System Security Command Reference guide](#) for more commands.

Enable Trust in Software

The second component in implementing a trustworthy system is to enable trust in software.

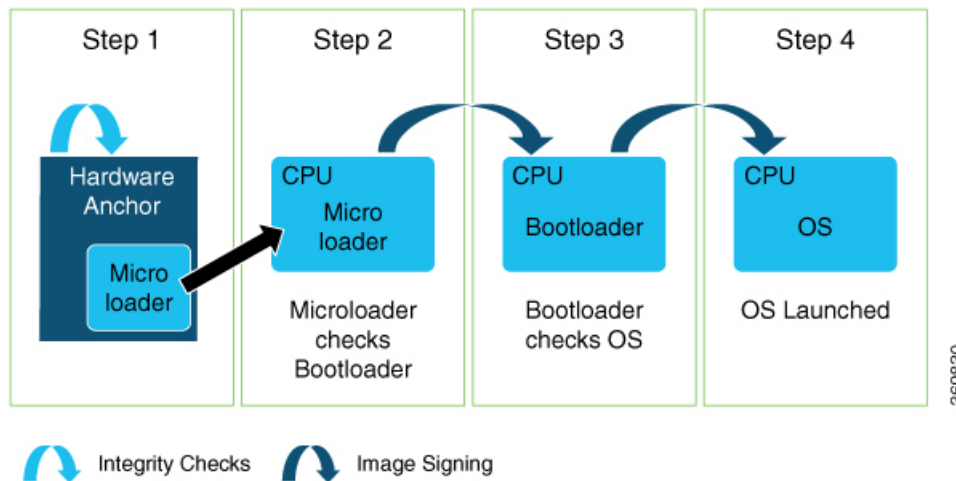
In Cisco IOS XR7, trust in the software is enabled through:

- Secure Boot
- Secure iPXE

Secure Boot

Cisco Secure Boot helps to ensure that the code that executes as part of the software image boot up on Cisco routers is authentic and unmodified. Cisco IOS XR7 platforms support the hardware-anchored secure boot which is based on the standard Unified Extensible Firmware Interface (UEFI). This UEFI-based secure boot protects the microloader (the first piece of code that boots) in tamper-resistant hardware, establishing a root of trust that helps prevent Cisco network devices from executing tainted network software.

Figure 8: Secure Boot



The intent of Secure Boot is to have a trust anchor module (TAM) in hardware that verifies the bootloader code. A fundamental feature of secure boot is the barrier it provides that makes it that it is extremely difficult or nearly impossible to bypass these hardware protections.

Secure boot ensures that the bootloader code is a genuine, unmodified Cisco piece of code and that code is capable of verifying the next piece of code that is loaded onto the system. It is enabled by default.

When secure boot authenticates the software as genuine Cisco in a Cisco device with the TAM, the operating system then queries the TAM to verify whether the hardware is authentic. It verifies by cryptographically checking the TAM for a secure unique device identifier (SUDI) that comes only from Cisco.

The SUDI is permanently programmed into the TAM and logged by Cisco during Cisco's closed, secured, and audited manufacturing processes.

Booting the System with Trusted Software

In Cisco IOS XR7, the router supports the UEFI-based secure boot with Cisco-signed boot artifact verification. The following takes place:

Step 1: At bootup, the system verifies every artifact using the keys in the TAM.

Step 2: The following packages are verified and executed:

- Bootloader (Grand Unified Bootloader (GRUB), GRUB configuration, Preboot eXecution Environment (PXE), netboot)
- Initial RAM disk (Initrd)
- Kernel (operating system)

Step 3: Kernel is launched.

Step 4: Init process is launched.

Step 5: All Cisco IOS XR RPMs are installed with signature verification.

Step 6: All required services are launched.

Secure iPXE – Secure Boot Over the Network

The iPXE server is an HTTP server discovered using DHCP that acts as an image repository server. Before downloading the image from the server, the Cisco router must authenticate the iPXE server.



Note A secure iPXE server must support HTTPS with self-signed certificates.

The Cisco router uses certificate-based authentication to authenticate the iPXE server. The router:

- Downloads the iPXE self-signed certificates
- Uses the Simple Certificate Enrollment Protocol (SCEP)
- Acquires the root certificate chain and checks if it's self-signed

The root certificate chain is used to authenticate the iPXE server. After successful authentication, a secure HTTPS channel is established between the Cisco router and the iPXE server. Bootstrapper protocol (Bootp), ISO, binaries, and scripts can now be downloaded on this secure channel.

Establish and Maintain Trust at Steady State

The third component in implementing a trustworthy system is to maintain trust in the steady or runtime state.

Attackers are seeking long-term compromise of systems and using effective techniques to compromise and persist within critical infrastructure devices. Hence, it is critical to establish and maintain trust within the network infrastructure devices at all points during the system runtime.

In Cisco IOS XR7, trust is established and maintained in a steady state through:

- SELinux
 - SELinux Policy
 - SeLinux Mode
- Secure Install
 - RPM Signing and Validation
 - Third-Party RPMs
- SSD Encryption

SELinux

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including mandatory access controls (MAC).

A kernel integrating SELinux enforces MAC policies that confine user programs and system servers to the minimum amount of privileges they require to do their jobs. This reduces or eliminates the ability of these programs and daemons to cause harm when compromised (for example, through buffer overflows or misconfigurations). This confinement mechanism operates independently of the traditional Linux access control mechanisms. SELinux has no concept of a "root" super-user and does not share the well-known shortcomings of the traditional Linux security mechanisms (such as a dependence on setuid/setgid binaries).

On Cisco IOS XR7 software, only Targeted SELinux policies are used, so that only third-party applications are affected by the policies; all Cisco IOS XR programs can run with full root permission.

With Targeted SELinux, using targeted policies, processes that are targeted run in a confined domain. For example, the httpd process runs in the httpd_t domain. If a confined process is compromised by an attacker, depending on the SELinux policy configuration, the attacker's access to resources and the possible damage that can result is limited.



Note Processes running in unconfined domains fall back to using discretionary access control (DAC) rules.

DAC is a type of access control defined as a means of restricting access to objects based on the identity of the subjects or the groups (or both) to which they belong.

SELinux Policy

Each Linux user is mapped to an SELinux user through an SELinux policy. This allows Linux users to inherit the restrictions placed on SELinux users.

If an unconfined Linux user executes an application, which an SELinux policy defines as an application that can transition from the unconfined_t domain to its own confined domain, the unconfined Linux user is subject to the restrictions of that confined domain. The security benefit is that, even though a Linux user is running in unconfined mode, the application remains confined. Therefore, the exploitation of a flaw in the application is limited by the policy.

A confined Linux user is restricted by a confined user domain against the unconfined_t domain. The SELinux policy can also define a transition from a confined user domain to its own target confined domain. In such a case, confined Linux users are subject to the restrictions of that target confined domain.

SELinux Mode

There are three SELinux modes:

- **Enforcing:** When SELinux is running in enforcing mode, it enforces the SELinux policy and denies access based on SELinux policy rules.
- **Permissive:** In permissive mode, the SELinux does not enforce policy, but logs any denials. Permissive mode is used for debugging and policy development. This is the default mode.
- **Disabled:** In disabled mode, no SELinux policy is loaded. The mode may be changed in the boot loader, SELinux config, or at runtime with **setenforce**.

Role of the SELinux Policy in Boot Process

SELinux plays an important role during system startup. Because all processes must be labeled with their proper domain, the init process performs essential actions early in the boot process that synchronize labeling and policy enforcement.

Secure Install

The Cisco IOS XR software is shipped as RPMs. Each RPM consists of one or more processes, libraries, and other files. An RPM represents a collection of software that performs a similar functionality; for example, packages of BGP, OSPF, as well as the Cisco IOS XR Infra libraries and processes.

RPMs can also be installed into the base Linux system outside the Cisco IOS XR domain; however, those RPMs must also be appropriately signed.

All RPMs shipped from Cisco are secured using digitally signed Cisco private keys.

There are three types of packages that can be installed:

- Packages shipped by Cisco (open source or proprietary)
- Customer packages that replace Cisco provided packages
- Customer packages that do not replace Cisco provided packages

RPM Signing and Validation

RPMs are signed using Cisco keys during the build process.

The install component of the Cisco IOS XR automatically performs various actions on the RPMs, such as verification, activation, deactivation, and removal. Many of these actions invoke the underlying DNF installer. During each of these actions, the DNF installer verifies the signature of the RPM to ensure that it operates on a legitimate package.

Cisco RPMs are signed using GPG keys. The RPM format has an area dedicated to hold the signature of the header and payload and these are verified and validated via DNF package managers.

Third-Party RPMs

The XR Install enforces signature validation using the ‘gpgcheck’ option of DNF. Thus, any Third-Party RPM packages installation fails if done through the XR Install (which uses the DNF).

SSD Encryption

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
Solid State Drive (SSD) Encryption	Release 7.3.1	<p>This feature enables trust and security in the system’s steady state by encrypting data at the disk level. The encrypted data can be accessed <i>only</i> with a specific key stored in the TAM.</p> <p>In this release, the feature is supported on:</p> <ul style="list-style-type: none"> • N540-28Z4C-SYS-A/D • N540X-16Z4G8Q2C-A/D • N540-12Z20G-SYS-A/D • N540X-12Z16G-SYS-A/D • N540X-6Z18G-SYS-A/D

Customers are concerned about the security of sensitive data present on persistent storage media. User passwords are limited in their capability to protect data against attackers who can bypass the software systems and directly access the storage media.

In this case, only encryption can guarantee data confidentiality.

Cisco IOS XR Software introduces SSD encryption that allows encrypting data at the disk level. SSD encryption also ensures that the encrypted data is specific to a system and is accessible *only* with a specific key to decrypt them.

Data that can be encrypted is sensitive information such as, topology data, configuration data, and so on.

Encryption is an automatic process and can be achieved through the following:

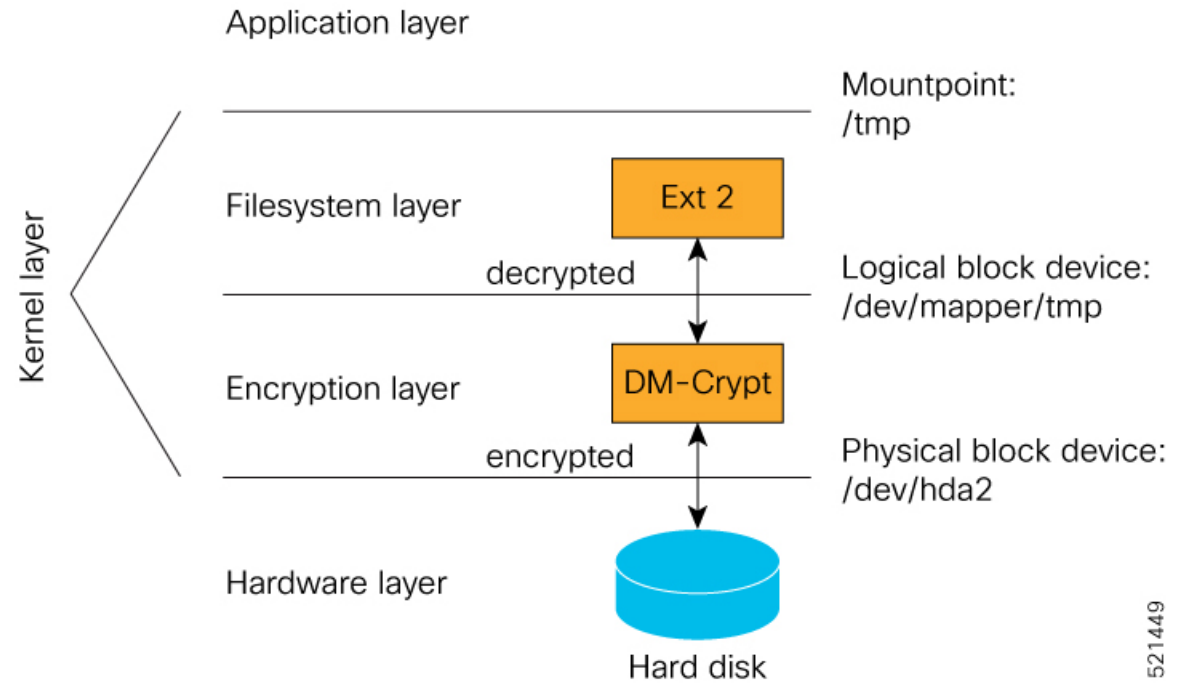
- DM-Crypt
- CPU with AES-NI support
- CryptSetup

DM-Crypt

DM-Crypt is a Linux kernel module that provides disk encryption. The module takes advantage of the Linux kernel’s device-mapper (DM) infrastructure. The DM provides a way to create virtual layers of block devices.

DM-crypt is a device-mapper target and provides transparent encryption of block devices using the kernel crypto API. Data written to the block device is encrypted; whereas, data to be read is decrypted. See the following figure.

Figure 9: DM-Crypt Encryption



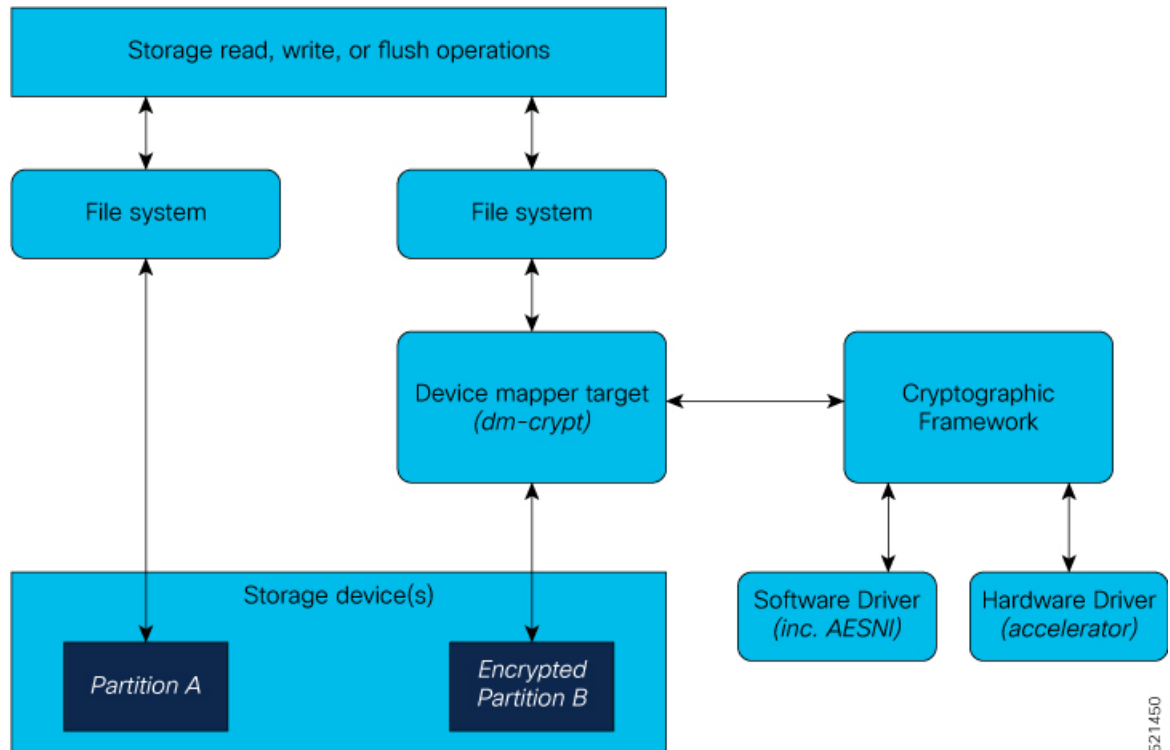
521449

AES-NI Support

Intel's Advanced Encryption Standard New Instructions (AES-NI) is a hardware-assisted engine that enables high-speed hardware encryption and decryption. This process leaves the CPU free to do other tasks.

When the input-output operations are started, the read-write requests that are directed at the encrypted block device are passed to the DM-Crypt. DM-Crypt then sends multiple cryptographic requests to the Cryptographic Framework. The crypto framework is designed to take advantage of off-chip hardware accelerators and provides software implementations when accelerators are not available. See the following image.

Figure 10: AES-NI Support



CryptSetup

DM-Crypt relies on user space tools, such as cryptsetup to set up cryptographic volumes. Cryptsetup is a command-line-interface (CLI) tool that interacts with DM-Crypt for creating, accessing, and managing encrypted devices.

Encrypted Logical Volume

An encrypted logical volume (LV) can be created during software installation on the following Cisco NCS 540 router variants:

- N540-28Z4C-SYS-A/D
- N540X-16Z4G8Q2C-A/D
- N540-12Z20G-SYS-A/D
- N540X-12Z16G-SYS-A/D
- N540X-6Z18G-SYS-A/D

You can activate or deactivate the encrypted disk partition on demand. In addition to being activated, all sensitive files are also migrated from the unencrypted disk partition to the encrypted disk partition. The encrypted files can be migrated back during deactivation.

You can activate the data encryption by using the `disk-encryption activate location` command. A sample output is as follows:

```
Router#disk-encryption activate location 0/RP0/CPU0
Tue Apr 16 14:35:00.939 UTC
```

Preparing system for backup. This may take a few minutes especially for large configurations.

```
Status report: node0_RP0_CPU0: START TO BACKUP
Router# Status report: node0_RP0_CPU0: BACKUP HAS COMPLETED SUCCESSFULLY
[Done]
```

The encrypted logical volume capacity is 150MB of disk space and is available as `/var/xr/enc` for applications to access.



Note Although applications can choose to use this space for storage, that data is not be part of the data migration if the software image is downgraded to a version that does not support encryption.

SSD Binding

When encryption is activated on a system, each card generates a random encryption key and stores it in its own secure storage—the Trust Anchor module (TAM). During successive reboots, the encryption key is read from the TAM and applied to unlock the encrypted device. Since each card stores its encryption key locally on the TAM, an SSD that is removed from one card and inserted into another cannot be unlocked by the key stored on that card, thereby making the SSD unusable.

If encryption is activated, the encrypted LV can only be unlocked by using the key stored in the TAM. So, if an encrypted SSD is removed and moved to another line card, the SSD cannot be unlocked. In other words, when you activate encryption, the SSD is bound to the card it is inserted in.

Data Zeroization

Zeroization refers to the process of deleting sensitive data from a cryptographic module.



Note In case of a Return Material Authorization (RMA), you must *factory reset* the data.

You can perform zeroization by using the `factory reset location` command from the XR prompt.



Caution Running this command while encryption is activated, deletes the master encryption key from the TAM and renders the motherboard unusable after the subsequent reload.

Boot Integrity and Trust Visibility

The secure boot first stage is rooted in the chip and all subsequent boot stages are anchored to the first successful boot. The system is, therefore, capable of measuring the integrity of the boot chain. The hash of each software boot image is recorded before it is launched. These integrity records are protected by the TAM. The boot chain integrity measurements are logged and these measurements are extended into the TAM.

Use the **Router#show platform security attest per 15 trustpoint ciscoaik nonce 4567** command to view the boot integrity and boot-chain measurements. Given below is a sample output:

```

RP/0/RP0/CPU0:ios# show platform security attest PCR 15 trustpoint CiscoAIK nonce 4567
location 0/RP0/CPU0

Sun Jun 21 03:07:18.394 UTC
Nonce: 4567

+-----+
| Node location: node0_RP0_CPU0 |
+-----+
Uptime: 495270
pcr-quote: /1RDR4AYACCBG/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAD8hUwAAAEf/////
AQAAACQAAAAALAAAAQALAwCAAAAGae1J8QIYe06nS2RUx0JYeoG8tM3bqeVdpW7COBwBt+g=
pcr-quote-signature:
EZbzSUGe89jSjH8ZqTgKJrZJBopEbd818C+h1Ec780qi7Li1WfCZQPIP6KCDV6HsRCVzLoFijgm1MLoZE2rakQq+/
1TgZOWSLjMY7RbjSFr8z/zbpVI+YLnOG+wytVYWuY33uKHBn/
YWokHwo+qVf7u9aLGhnrXKvRUaFknBiZtQGiyAdis6GbPTToqn0WSNly6DPh4UHZj1vLVwJsI48mbQUrAyCZrz/
XBHLM38tVJjqSrC0jw/6LF2DDoT5ks0VUFT7sqbysw4F56y+z/I1DBrrRW3GFOY46MOxDxLwS11/
n6zdoVjIKKeqKOnmhpBh72bJQAdeu/GVOYTrOSy4Q==
pcr-index      pcr-value
  15      oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=

RP/0/RP0/CPU0:ios# show platform security integrity hardware digest-algorithm SHA1 trustpoint
CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:14.594 UTC
Nonce: 4567

+-----+
| Node location: node0_RP0_CPU0 |
+-----+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495385
Known-good-digests:
Index  value
  0      3TDUS9iUDCFX3VkiCcOnySOQTPA=
observed-digests:
Index  value
  0      3TDUS9iUDCFX3VkiCcOnySOQTPA=
PCRs:
Index  value
  15     1Y3uKqNv1UJQUNZQxmZkiuG4blk=

RP/0/RP0/CPU0:ios# show platform security integrity hardware digest-algorithm SHA256
trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:31.110 UTC
Nonce: 4567

+-----+
| Node location: node0_RP0_CPU0 |
+-----+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495401
Known-good-digests:
Index  value
  0      3TDUS9iUDCFX3VkiCcOnySOQTPA=
observed-digests:
Index  value
  0      3TDUS9iUDCFX3VkiCcOnySOQTPA=
PCRs:
Index  value
  15     1Y3uKqNv1UJQUNZQxmZkiuG4blk=

RP/0/RP0/CPU0:ios# show platform security integrity hardware digest-algorithm SHA256

```



```

trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:43.782 UTC
Nonce: 4567

+-----+
Node location: node0_RP0_CPU0
+-----+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495414
Known-good-digests:
Index value
0 y3n/SsvyNb8g3o7FFRGCZwfb8EGxvMZg/PeN0NA71k=
observed-digests:
Index value
0 y3n/SsvyNb8g3o7FFRGCZwfb8EGxvMZg/PeN0NA71k=
PCRs:
Index value
15 oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=
Cisco AIK Certificate used for signing PCR
pcr-quote: /1RDR4AYACCBG/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hywAAAEf////
/AQAAACQAAAAALAAAAQALAwCAAAAGaelJ8QIYe06nS2RUx0JYeoG8tm3bqeVdpW7CObwBt+g=
pcr-quote-signature:
qyKbK7ndJbrgxeVnOodLWQzT7++NzrxJ9ERRvJzvTe4+8r6p0HGSEpHUhZHkXw4DbniHAK0Cs3dwg/
hGKGe4M8Lz+/k682yIjaFYyip0DHMaV2ny/1T7RSqm/6u3j/JZrZv39MaeHa3MyjjonzRf9oe7EBSFAKsa/D54eTR0eFtaxFy/
XdtM0VVQe2JRdoBVxnIBLGiVmGR1VVlmHvwwgX11AN6e3/soC1Vk3I5gjLldPHUYuJ/
7PTGyAwZsbdeigx8d4ViUUjSMzK7JISwXa8k4GiPQVLBhtqR+RA9scmMZTbKLSG3luIWKQeyCtXMYE1VOeW8WQ1AvioMICw==
RP/0/RP0/CPU0:ios#show platform security integrity hardware digest-algorithm$
Sun Jun 21 03:09:56.794 UTC
Nonce: 4567

+-----+
Node location: node0_RP0_CPU0
+-----+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495427
Known-good-digests:
Index value
0 3TDUS9iUDCFX3VkiCconySOQTPA=
observed-digests:
Index value
0 3TDUS9iUDCFX3VkiCconySOQTPA=
PCRs:
Index value
15 1Y3uKqNv1UJQUNZQxmZkiuG4blk=

RP/0/RP0/CPU0:ios#

```

You can also use Cisco-IOS-XR-remote-attestation-act.yang to fetch the boot integrity over the NETCONF protocol.

The command displays both, the integrity log values and the assurance that these values have not been tampered. These measurements include the following parameters:

- Micro loader hash
- Boot loader hash
- Image signing and management key hashes
- Operating system image hash

```

platform-pid string Platform ID
Event log [key: event_number]: Ordered list of TCG described event log

```

```

                                that extended the PCRs in the order they
                                were logged
+-- event_number  uint32 Unique event number of this even
+-- event_type    uint32 log event type
+-- PCR_index     uint16 PCR index that this event extended
+-- digest        hex-string The hash of the event data
+-- event_size    uint32 Size of the event data
+-- event_data    uint8[] event data, size determined by event_size
PCR [index] - List of relevant PCR contents
+-- index         uint16 PCR register number
+-- value         uint8[] 32 bytes - PCR register content
PCR Quote binary TPM 2.0 PCR Quote
PCR Quote Signature binary Signature of the PCR quote using TAM-held ECC or RSA restricted
key with the optional nonce if supplied

```

**Note**

- Platform Configuration Register (PCR) 0-9 are used for secure boot.
- Signature version designates the format of the signed data.
- The signature digest is SHA256.
- The signing key is in a Trusted Computing Group (TCG) compliant format.



Note

Use the **show platform security tam** command to view the TAM device details. The following example shows a truncated output of the command:

```
Router#show platform security tam all location all
Mon Apr 15 14:42:34.649 UTC
-----
Node - node0_RP0_CPU0
-----
Device Type           -      AIKIDO Extended
Device PID            -      N540X-12Z16G-SYS-A
Device Serial Number  -      FOC2333NJ0J
Device Firmware Version - 0x24.000b
Server Version        -      3
Server Package Version - 9.4.1
Client Package Version - 9.4.1

Sudi Root Cert:
-----
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      01:9a:33:58:78:ce:16:c1:c1
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: O=Cisco, CN=Cisco Root CA 2099
    Validity
      Not Before: Aug  9 20:58:28 2016 GMT
      Not After  : Aug  9 20:58:28 2099 GMT
    Subject: O=Cisco, CN=Cisco Root CA 2099
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:d3:b6:e3:35:7e:0d:3e:f4:67:e5:8a:4e:1a:c6:
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Subject Key Identifier:
        38:95:57:0F:34:23:4E:F3:A1:26:20:BA:14:91:C7:41:88:1D:A3:5B
    Signature Algorithm: sha256WithRSAEncryption
      8d:e2:99:a3:ee:31:77:4e:53:16:da:bd:f6:72:a7:58:0d:09:

Sudi Sub CA Cert:
-----
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      0a:64:75:52:4c:d8:61:7c:62
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: O=Cisco, CN=Cisco Root CA 2099
    Validity
      Not Before: Aug 11 20:28:08 2016 GMT
      Not After  : Aug  9 20:58:27 2099 GMT
    Subject: CN=High Assurance SUDI CA, O=Cisco
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:bd:dc:de:49:67:43:23:a9:51:64:36:11:bc:0e:
```

```

    Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
  X509v3 Basic Constraints: critical
    CA:TRUE, pathlen:0
  Authority Information Access:
    CA Issuers - URI:https://www.cisco.com/security/pki/certs/crca2099.cer
    OCSP - URI:http://pkicvs.cisco.com/pki/ocsp

  X509v3 Authority Key Identifier:
    keyid:38:95:57:0F:34:23:4E:F3:A1:26:20:BA:14:91:C7:41:88:1D:A3:5B

  X509v3 Certificate Policies:
    Policy: 1.3.6.1.4.1.9.21.1.30.0
    CPS: http://www.cisco.com/security/pki/policies/

  X509v3 CRL Distribution Points:

    Full Name:
      URI:http://www.cisco.com/security/pki/crl/crca2099.crl

  X509v3 Subject Key Identifier:
    EA:6B:A3:B9:C1:13:97:7E:1B:FB:3A:8D:68:60:07:39:5F:87:48:FA
Signature Algorithm: sha256WithRSAEncryption
5c:a9:81:0e:80:01:e1:19:62:a7:77:03:3d:d3:55:d7:d8:49:

Sudi Cert:
-----
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 29200071 (0x1bd8ec7)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=High Assurance SUDI CA, O=Cisco
    Validity
      Not Before: Sep  5 03:39:36 2019 GMT
      Not After : Aug  9 20:58:26 2099 GMT
    Subject: serialNumber=PID:N540X-12Z16G-SYS-A SN:FOC2333NJ0J, O=Cisco, OU=ACT-2 Lite
SUDI, CN=Cisco NCS 540 System with 12x10G+4x1G Cu+12x1G AC Chassis
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:ca:2a:8a:b4:87:8b:43:68:17:d3:b2:43:44:ca:

        Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Key Usage: critical
    Digital Signature, Non Repudiation, Key Encipherment
  X509v3 Basic Constraints: critical
    CA:FALSE
  X509v3 Subject Alternative Name:
    0...N.
+.....@917C927B4B340B908703945A7A0A6D14D0207ADB2FB622DFA8C83538FD7E63B5.
B..+.....5.3ChipID=QvZQd9q9psveoAz6QJQeNAAAAAAAAAAAAAAAAAAAAAAAA=
    Signature Algorithm: sha256WithRSAEncryption
    5b:67:da:2e:e5:d4:07:f2:ff:9c:17:c9:54:78:8b:da:16:df:

```

The boot integrity verification is automatic and the BIOS reports the values to the PCR. The boot integrity verification process consists of the following steps:

1. Report Boot 0 version and look up the expected integrity value for this platform and version.

2. Report bootloader version and look up the expected integrity value for this platform and version.
3. Report OS version and look up the expected integrity value for this platform and version.
4. Using the integrity values obtained from steps 1-3, compute the expected PCR 0 and PCR 8 values
5. Compare the expected PCR values against the actual PCR values.
6. Verify the nonced signature to ensure the liveness of the response data (this assumes unique nonced are being passed). Note that this signature verification must be performed only with the platform identity verified using SUDI.
7. (Optional) Verify the software image (IOS XR) version is with what is expected to be installed on this platform.

A failure of any of the above steps indicates either a compromised system or an incomplete integrity value database.

Secure gRPC

gRPC (gRPC Remote Procedure Calls) is an open source remote procedure call (RPC) system that provides features such as, authentication, bidirectional streaming and flow control, blocking or nonblocking bindings, and cancellation and timeouts. For more information, see <https://opensource.google.com/projects/grpc>.

TLS (Transport Layer Security) is a cryptographic protocol that provides end-to-end communications security over networks. It prevents eavesdropping, tampering, and message forgery.

In Cisco IOS XR7, by default, TLS is enabled in gRPC to provide a secure connection between the client and server.

Integrity Measurement Architecture (IMA)

The goals of the Linux kernel integrity subsystem are to:

- detect whether files are accidentally or maliciously altered, both remotely and locally
- measure the file by calculating the hash of the file content
- appraise a file's measurement against a known good value stored as an extended attribute
- enforce local file integrity

There are three components in the Linux kernel integrity subsystem:

- IMA Measurement
- IMA Appraisal
- IMA Audit



Note These goals are complementary to the Mandatory Access Control (MAC) protections provided by SELinux.

IMA Measurement

IMA maintains a runtime measurement list and—because it is also anchored in the hardware Trusted Anchor module (TAM)—an aggregate integrity value over this list. The benefit of anchoring the aggregate integrity value in the TAM is that the measurement list cannot be compromised by any software attack without being detectable. As a result, on a trusted boot system, IMA-measurement can be used to attest to the system's runtime integrity.

For more information about IMA, download the IMA whitepaper, [An Overview of The Linux Integrity Subsystem](#).

IMA Audit

IMA audit generates an event log every time it finds a file opened for reading or executing that has a mismatch between the measured file hash and the one stored in the extended attribute.

This data integrity verification event is recorded in the audit log.

There are three reasons an integrity log is recorded in the audit log – invalid signature, invalid hash and missing hash. The audit log has the following key information:

- type - INTEGRITY_DATA - Triggered to record a data integrity verification event run by the kernel.
- pid - Process ID of the calling process that opened the file with integrity verification failure.
- subject - SELinux file context label. SELinux runs in Permissive mode. Any access control violation is only logged in the audit log and the application is still allowed to run.
- op - Operation (appraise_data).
- cause - Reason for integrity verification failure (invalid-signature, invalid-hash, missing-hash).
- comm - Calling process.
- name – Name of the file with full path that was appraised.

The following output shows an instance where the IMA appraisal causes the execution of a tampered binary executable to fail. The integrity violation logged is **Invalid Signature**, the integrity violation log type is **Integrity Data**, and the appraised file is **/usr/bin/zip**.

```
RP/0/RP0/CPU0:NCS-540-C-LNT#run
Mon Apr 29 08:39:26.793 UTC
[node0_RP0_CPU0:~]$cat /var/log/audit/audit.log | grep -i integ | grep zip | fold -w 100
type=INTEGRITY_DATA msg=audit(1714378019.193:866): pid=2236 uid=0 auid=4294967295
ses=4294967295 sub
j=iosxradmin_u:iosxradmin_r:iosxradmin_t:s0 op="appraise_data" cause="invalid-signature"
comm="sh" n
ame="/usr/bin/zip" dev="dm-14" ino=147881 res=0
type=INTEGRITY_DATA msg=audit(1714378019.197:867): pid=2236 uid=0 auid=4294967295
ses=4294967295 sub
j=iosxradmin_u:iosxradmin_r:iosxradmin_t:s0 op="appraise_data" cause="invalid-signature"
comm="sh" n
ame="/usr/bin/zip" dev="dm-14" ino=147881 res=0
```

The following output shows an instance when an audit log was recorded because the file was missing an IMA signature and was opened for either reading or execution. This resulted in a “missing-hash” event log.

```
[node0_RP0_CPU0:/ima-appraisal]$zip --version | head -2
sh: /usr/bin/zip: Permission denied
[node0_RP0_CPU0:/ima-appraisal]$
[node0_RP0_CPU0:/ima-appraisal]$cat /var/log/audit/audit.log | grep -i integ | fold -w 100
type=INTEGRITY_DATA msg=audit(1714500558.187:556): pid=52560 uid=0 auid=4294967295
```

```
ses=4294967295 su
bj=iosxradmin_u:iosxradmin_r:iosxradmin_t:s0 op=appraise_data cause=missing-hash comm="sh"
name="/us
r/bin/zip" dev="dm-11" ino=1507384 res=0
```

IMA Policy

The IMA policy is not user-defined and is created by default. It contains a policy rule set that defines exactly which files on the file system should be measured or appraised.

Each policy rule must start with one of the following directives:

- `measure`: Perform IMA measurement
- `dont_measure`: Exclude from IMA measurement
- `appraise`: Perform IMA appraisal
- `dont_appraise`: Exclude from IMA appraisal



Note IMA policy is protected at runtime – it cannot be read or modified.

Verifying the IMA Appraisal “enforce” Mode

To display the content of the IMA appraisal mode, query the kernel command line and look for “`ima_appraise=enforce`”.

```
$ cat /proc/cmdline
```

To query the content of the IMA measurement logs:

```
$ cat /sys/kernel/security/ima/ascii_runtime_measurements
```

To display the total number of files measured:

```
$ cat /sys/kernel/security/ima/runtime_measurements_count
```

To display the total number of integrity violations:

```
$ cat /sys/kernel/security/ima/violations
```

To access other user space interfaces in sysfs that are specific to the `cisco_ima` measurement:

```
$ ls /sys/kernel/security/cisco_ima
```

IMA Signatures

The IMA appraisal provides local integrity, validation, and enforcement of the measurement against a known good value stored as an extended attribute—`security.ima`. The method for validating file data integrity is based on a digital signature, which in addition to providing file data integrity also provides authenticity. Each file (RPM) shipped in the image is signed by Cisco during the build and packaging process and validated at runtime using the IMA public certificate stored in the TAM.

All RPMs contain Cisco IMA signatures of the files packaged in the RPM, which are embedded in the RPM header. The IMA signature of the individual file is stored in its extended attribute during RPM installation. This protects against modification of the Cisco RPMs.

The IMA signature format used for IMA can have multiple lines and every line has comma-separated fields. Each line entry will have the filename, hash, and signature as illustrated below.

- File – Filename with the full path of the file hashed and signed
- Hash – SHA256 hash of the file
- Signature – RSA2048 key-based signature

How Trustworthiness is Implemented

The following sequence of events takes place automatically when the Cisco routers that support the IOS XR7 operating system are powered up:

1. At power UP, the micro-loader in the chip verifies the digital signature of BIOS using the keys stored in the Trusted Anchor module (TAm). The BIOS signature verification is logged and the measurement is extended into a PCR.
2. The BIOS then verifies the signature of the boot-loader using keys stored in TAm. The boot-loader signature verification is logged and the measurement is extended into the PCR.
3. If the validation is successful, the BIOS launches the bootloader. The bootloader uses the keys loaded by the BIOS to verify the sanctity of the kernel, initial RAM disk (initrd) file system, and grub-config file. Each verification operation is logged, and the PCR in TAm is extended.
4. The initrd is loaded to create the initial file system.
5. The kernel is launched and the kernel keyrings are populated with the appropriate keys from the TAm.
6. The init process is launched. Whenever an executable or a shared library is invoked, the IMA kernel hook validates the signature using the certificates in IMA keyring, which is then used to validate the signature attached to the file.
7. The Cisco IOS XR7 RPM is installed with the signed verification. The results of RPM verification are logged.
8. Cisco IOS XR7 processes are launched with IMA measurement.
9. TAm services are launched.
10. Cisco IOS XR7 application runs the initial admin user configuration and stores the credentials into TAm secure storage.

Manual provisioning of user credentials is now complete.

The Cisco routers perform the above steps, which is a holistic approach to integrate trust. Trust begins in hardware, next the system verifies the trustworthiness of the network operating system, after bootup, the system maintains trust at runtime, last, the system visualizes and reports on trust. You can verify the boot status by executing the following command:

```
Router#show platform security integrity log secure-boot
Fri Apr 12 17:13:43.867 UTC

+-----+
Node location: node0_RP0_CPU0
+-----+
Secure Boot Status: Enabled
```

Understanding Key Concepts in Security

Attestation

Attestation is a mechanism used to attest the software's integrity. The verifier trusts that the attested data is accurate because it is signed by a TPM whose key is certified by the CA.

Attestation Identity Key

An Attestation Identity Key (AIK) is a restricted key that is used for signing attestation requests.

Bootloader

The bootloader is a piece of code that runs before any operating system begins to run. Bootloaders contain several ways to boot the OS kernel and also contain commands for debugging and modifying the kernel environment.

Certificates and Keys in TAM

All database keys are signed by the KEK. Any update to the keys requires the KEK or PK to sign in, using time-based authentic variables. Some of the keys on the database are:

- Image signing certificate: This is the X.509 certificate corresponding to the public key and is used for validating the signature of grub, initrd, kernel, and kernel modules.
- IOS-XR Key: A public key certificate signed by the KEK. This key is common to all Cisco NCS 540 Series routers and is used to sign GRUB, initrd, kernel and kernel modules.
- RPM key: Used for signing RPMs.
- IMA public key certificate: Used for Integrity Measurement Architecture (IMA), and used to validate the IMA signature of the files.
- BIOS or Firmware Capsule Update key: Used to sign the outer capsule for BIOS or firmware updates. It is the same as the secure boot key.
- Platform key (PK) and Key Enrollment Key (KEK): These are public keys and certificates used to manage other keys in the TAM.
- LDWM Key: In the Cisco IOS XR7, the LDWM key is stored in the hardware trust anchor module and is used for validating the BIOS.

Golden ISO (GISO)

A GISO image includes a base binary artifact (an ISO) for the Linux distribution that is used on the server fleet, packages, and configuration files that can be used as a base across all servers.

The GISO image for Cisco IOS XR7 software contains the IOS XR RPMs and third-party RPMs.

GRand Unified Bootloader (GRUB)

GNU GRUB (or just GRUB) is a boot loader package that loads the kernel and supports multiple operating systems on a device. It is the first software that starts at a system boot.

Hash Function

A hash function is any function that is used to map data of arbitrary size onto data of a fixed size.

Initramfs

Initramfs, a complete set of directories on a normal root filesystem, is bundled into a single cpio archive and compressed with one of the several compression algorithms. At boot time, the boot loader loads the kernel and the initramfs image into memory and starts the kernel.

initrd

initial RAM disk is an initial root file system that is mounted before the real root file system is made available. The initrd is bound to the kernel and loaded as part of the kernel boot procedure.

JTAG

JTAG is a common hardware interface that provides a system with a way to communicate directly with the chips on a board. JTAG is used for debugging, programming, and testing on embedded devices.

Nonce Value

A nonce value is an arbitrary number that can be used only once in a cryptographic communication. It is a random or pseudo-random number that is issued in an authentication protocol to ensure that the old communications are not reused in replay attacks.

Platform Configuration Register (PCR)

A PCR is a shielded register or memory region large enough to hold the contents of a hash operation. A PCR is initialized to a well-known value at power-up, and typically cannot be reset.

PCR Extend

The only way to change the value held in a PCR is to perform an “extend” operation, which is defined as:

```
PCR[x]new = hash ( PCR[x]old || hash ( measurement value ) )
```

Trust Anchor module (TAM)

The Cisco Trust Anchor module (TAM) helps verify that Cisco hardware is authentic and provides additional security services.

Trusted Platform Module (TPM)

A Trusted Platform Module (TPM) is a specialized chip on an endpoint device that stores RSA encryption keys specific to the host system for hardware authentication. This key pair is generated by the TPM based on the Endorsement Key and an owner-specified password.

Root of Trust for Storage

TPM 2.0-compliant Platform Configuration Registers (PCRs) form the Root of Trust for Storage.

