System Security Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.9.x

First Published: 2023-03-01

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

CONTENTS

PREFACE

Preface xiii

Changes to This Document xiii

Communications, Services, and Additional Information xiii

CHAPTER 1

New and Changed Feature Information 1

System Security Features Added or Modified in IOS XR Release 7.9.x 1

CHAPTER 2

YANG Data Models for System Security Features 3

Using YANG Data Models 3

CHAPTER 3

Implementing Trustworthy Systems 5

Need for Trustworthy Systems 5

Enable Trust in Hardware 6

Enable Trust in Hardware 6

Hardware Integrity Check Using Chip Guard Functionality 7

Attestation 8

Enable Trust in Software 9

Secure Boot 10

Secure iPXE – Secure Boot Over the Network 11

Verify Secure Boot Status 12

Establish and Maintain Trust at Steady State 12

SELinux 12

SELinux Policy 13

SELinux Mode 13

Role of the SELinux Policy in Boot Process 13

Secure Install 14

RPM Signing and Validation 14
Third-Party RPMs 14
SSD Encryption 14
DM-Crypt 15
AES-NI Support 15
CryptSetup 16
Encrypted Logical Volume 16
SSD Binding 17
Data Zeroization 17
Runtime Defences (RTD) 17
Address Space Layout Randomization (ASLR) 17
Kernel Address Space Layout Randomization (KASLR) 18
Built-in Object Size Checker (BOSC) 18
Executable Space Protection (XSpace) 18
RTD Monitor 18
Boot Integrity and Trust Visibility 18
Secure gRPC 25
Integrity Measurement Architecture (IMA) 25
IMA Signatures 26
How Trustworthiness is Implemented 26
Understanding Key Concepts in Security 27

CHAPTER 4 **Configuring AAA Services** 31
Prerequisites for Configuring AAA Services 31
Restrictions for Configuring AAA Services 31
Information About Configuring AAA Services 32
User, User Groups, and Task Groups 32
User Categories 32
User Groups 32
Task Groups 33
Administrative Model 34
Administrative Access 34
AAA Database 35
Remote AAA Configuration 35

AAA Configuration 36
Authentication 37
Password Types 39
Type 8 and Type 9 Encryption Methods 39
Type 10 Password Encryption for User Management 40
AAA Password Security for FIPS Compliance 40
AAA Password Security Policies 40
Minimum Password Length for First User Creation 43
Password Policy for User Secret 43
Task-based Authorization 43
Task IDs 44
General Usage Guidelines for Task IDs 44
Task IDs for TACACS+ and RADIUS Authenticated Users 45
Task Maps 45
Privilege Level Mapping 47
XML Schema for AAA Services 47
Netconf and Restconf for AAA Services 47
About RADIUS 48
Network Security Situations in Which RADIUS is Unsuitable 49
RADIUS Operation 49
Differentiated Services Code Point (DSCP) Marking Support for TACACS Packets 50
How to Configure AAA Services 50
Configure Task group 50
Task Group Configuration 52
Configure User Groups 53
Configure First User on Cisco Routers 54
Configure Users 56
Password Masking For Type 7 Password Authentication 58
Configure Type 8 and Type 9 Passwords 59
Configure Type 10 Password Encryption 60
Configure AAA Password Policy 61
Configure Password Policy for User Secret and Password 63
Configure Router to RADIUS Server Communication 66
Configure RADIUS Dead-Server Detection 69

Configure Per VRF AAA 71
New Vendor-Specific Attributes (VSAs) 71
Configure TACACS+ Server 73
Configure Authorization for a TACACS+ Server 76
Configure Authentication for a TACACS+ Server 77
Configure Accounting for a TACACS+ Server 78
Configure RADIUS Server Groups 79
Configure TACACS+ Server Groups 80
Configure Per VRF TACACS+ Server Groups 82
Configure AAA Method Lists 83
Configuring Authentication Method Lists 84
Configuring Authorization Method Lists 86
Configuring Accounting Method Lists 89
Generate Interim Accounting Records 91
Applying Method Lists for Applications 92
Enabling AAA Authorization 92
Enable Accounting Services 94
Configure Login Parameters 95
Command Accounting 96
Command Authorization Using Local User Account 97
Configure Command Authorization Using Local User Account 99
Feature Behavior and Use Case Scenarios 100
Configuration Example for AAA Services 102

CHAPTER 5 **Implementing Certification Authority Interoperability** 105
Implementing Certification Authority Interoperability 105
Prerequisites for Implementing Certification Authority 106
How to Implement CA Interoperability 106
Configure Router Hostname and IP Domain Name 106
Generate RSA Key Pair 108
Import Public Key to the Router 110
Declare Certification Authority and Configure Trusted Point 112
Authenticate CA 114
Request Your Own Certificates 114

CA enrollment URL 115
Configure Certificate Enrollment Using Cut-and-Paste 116
Certificate Authority Trust Pool Management 119
CA Certificate Bundling in the Trust Pool 120
Prerequisites for CA Trust Pool Management 120
Restrictions for CA trust pool management 120
Updating the CA Trustpool 120
Retrieve CRL through the HTTP Proxy Server 122
Configuring Optional Trustpool Policy Parameters 124
Handling of CA Certificates appearing both in Trust Pool and Trust Point 124
Expiry Notification for PKI Certificate 125
Learn About the PKI Alert Notification 125
Restrictions for PKI Credentials Expiry Alerts 126
Regenerate the Certificate 126
Integrating Cisco IOS XR and Crosswork Trust Insights 127
How to Integrate Cisco IOS XR and Crosswork Trust Insights 128
Support for Ed25519 Public-Key Signature System 142
Generate Crypto Key for Ed25519 Signature Algorithm 142
Integrate Cisco IOS XR with Cisco Crosswork Trust Insights using Ed25519 143
Public Key-Pair Generation in XR Config Mode 144
Information About Implementing Certification Authority 147
Supported Standards for Certification Authority Interoperability 147
Certification Authorities 147
Purpose of CAs 147
CA Registration Authorities 148

CHAPTER 6 **Implementing Keychain Management 149**
Prerequisites for Configuring Keychain Management 149
Restrictions for Implementing Keychain Management 149
Information About Implementing Keychain Management 149
Lifetime of Key 150
How to Implement Keychain Management 150
Configure Keychain 150
Configure Tolerance Specification to Accept Keys 152

Configure Key Identifier for Keychain 152
Configure Text for Key String 153
Determine Valid Keys 154
Configure Keys to Generate Authentication Digest for Outbound Application Traffic 155
Configure Cryptographic Algorithm 156
Configuration Examples for Implementing Keychain Management 158
Configuring Keychain Management: Example 158

CHAPTER 7 **Configuring MACsec** 159
Understanding MACsec Encryption 159
MKA Authentication Process 160
MACsec Frame Format 161
Advantages of Using MACsec Encryption 161
Hardware Support for MACsec 162
MACsec PSK 162
Fallback PSK 163
Configuring and Verifying MACsec Encryption 164
Create a MACsec keychain 166
Securing the MACsec Pre-shared Key (PSK) Using Type 6 Password Encryption 168
Configuring a Primary Key and Enabling the Type 6 Password Encryption Feature 168
Configuring MACsec Pre-shared Key (PSK) For Type 6 Password Encryption 169
Create a user-defined MACsec policy 170
MACsec SAK Rekey Interval 171
EAPoL Ether-type and destination address 172
Applying MACsec Configuration on an Interface 173
Configure EAPoL Ether-type 0x876F 179
Configure EAPoL destination broadcast address 180
Configure EAPoL destination bridge group address 181
MACsec mode on PHY 182
Enable MACsec Mode on PHY 182
MACsec Policy Exceptions 183
How to Create MACsec Policy Exception 184
Verifying MACsec Encryption on IOS XR 185
Verifying MACsec Encryption on Cisco 8000 Series Routers 195

MACsec SecY Statistics 199
Querying SNMP Statistics Using CLI 199
MACsec SNMP MIB (IEEE8021-SECY-MIB) 200
secyIfTable 201
secyTxSCTable 202
secyTxSATable 202
secyRxSCTable 203
secyRxSATable 203
secyCipherSuiteTable 204
secyTxSAStatsTable 204
secyTxSCStatsTable 204
secyRxSAStatsTable 205
secyRxSCStatsTable 205
secyStatsTable 206
Obtaining the MACsec Controlled Port Interface Index 206
SNMP Query Examples 207
Power-on Self-Test KAT for Common Criteria and FIPS 207
How to Configure Power-on Self-Test KAT 207
Secure Key Integration Protocol 210
Configuring Point to Point MACsec Link Encryption using SKIP 212
Related Commands for MACsec 217

CHAPTER 8 **MACSec Using EAP-TLS Authentication 219**
Guidelines and Limitations for EAP-TLS Authentication 219
IEEE 802.1X Device Roles 220
Prerequisites for MACSec MKA Using EAP-TLS Authentication 220
MACsec with Local EAP-TLS Authentication 220
Configure MACSec Encryption Using EAP-TLS Authentication 220
Configure RADIUS Server 220
Configure 802.1X Authentication Method 221
Generate RSA Key Pair 221
Configure Trustpoint 222
Configure Domain Name 223
Authenticate Certificate Authority and Request Certificates 223

Configure EAP Profile 224
Configure 802.1X Profile on the Device 224
Configure MACSec EAP and 802.1X Profile on an Interface 225
Verify MACSec EAP and 802.1X Configuration on Interface 225

CHAPTER 9 **Implementing URPF 229**
Understanding uRPF 229
Configuring uRPF in Loose Mode 230
Configuring uRPF in Strict Mode 232

CHAPTER 10 **Implementing Type 6 Password Encryption 235**
How to Implement Type 6 Password Encryption 235
Enabling Type6 Feature and Creating a Primary Key (Type 6 Server) 235
Implementing Key Chain for BGP Sessions (Type 6 Client) 238
Creating a BGP Session (Type 6 Password Encryption Use Case) 239

CHAPTER 11 **Implementing Management Plane Protection 241**
Prerequisites for Implementing Management Plane Protection 241
Restrictions for Implementing Management Plane Protection 241
Information About Implementing Management Plane Protection 242
Inband Management Interface 242
Out-of-Band Management Interface 242
Peer-Filtering on Interfaces 242
Control Plane Protection Overview 242
Management Plane 243
Management Plane Protection Feature 243
Benefits of the Management Plane Protection Feature 244
How to Configure a Device for Management Plane Protection 244
Configuring a Device for Management Plane Protection for an Inband Interface 244
Configuring a Device for Management Plane Protection for an Out-of-band Interface 245
Configuration Examples for Implementing Management Plane Protection 245
Configuring Management Plane Protection: Example 246
Additional References 247

CHAPTER 12 **Implementing Secure Shell** 249

Information About Implementing Secure Shell 250

SSH Server 250

SSH Client 250

SFTP Feature Overview 251

RSA Based Host Authentication 252

RSA Based User Authentication 252

SSHv2 Client Keyboard-Interactive Authentication 253

SSH and SFTP in Baseline Cisco IOS XR Software Image 253

CiscoSSH 254

Guidelines for Using CiscoSSH 256

Prerequisites for Implementing Secure Shell 260

Guidelines and Restrictions for Implementing Secure Shell 261

How to Implement Secure Shell 262

Configure SSH 262

Automatic Generation of SSH Host-Key Pairs 265

Configure the Allowed SSH Host-Key Pair Algorithms 265

Ed25519 Public-Key Signature Algorithm Support for SSH 267

How to Generate Ed25519 Public Key for SSH 268

Configure the SSH Client 268

Order of SSH Client Authentication Methods 270

How to Set the Order of Authentication Methods for SSH Clients 270

Configure Secure Shell: Example 271

Multi-channeling in SSH 271

Restrictions for Multi-channeling Over SSH 271

Client and Server Interaction Over Multichannel Connection 271

Configure Client for Multiplexing 272

SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm 273

Disable HMAC Algorithm 274

Enable Cipher Public Key 274

User Configurable Maximum Authentication Attempts for SSH 276

Configure Maximum Authentication Attempts for SSH 277

X.509v3 Certificate-based Authentication for SSH 278

Configure X.509v3 Certificate-based Authentication for SSH 281
Selective Authentication Methods for SSH Server 286
Disable SSH Server Authentication Methods 286
SSH Port Forwarding 287
How to Enable SSH Port Forwarding 289

CHAPTER 13 **Configuring FIPS Mode** 293
Prerequisites for Configuring FIPS 294
How to Configure FIPS 294
Enable FIPS mode 295
Configure FIPS-compliant Keys 296
Configure FIPS-compliant Key Chain 299
Configure FIPS-compliant Certificates 300
Configure FIPS-compliant OSPFv3 301
Configure FIPS-compliant SNMPv3 Server 303
Configure FIPS-compliant SSH Client and Server 303

CHAPTER 14 **Implementing Secure Logging** 307
System Logging over Transport Layer Security (TLS) 307
Restrictions for Syslogs over TLS 309
Configuring Syslogs over TLS 309

CHAPTER 15 **Implementing MAC Authentication Bypass** 313
MAC Authentication Bypass 314
Configure MAC Authentication Bypass 316

CHAPTER 16 **Cisco MASA Service** 321
Why Do I Need Cisco MASA? 322
Use Cases for Ownership Vouchers 322
Authentication Flow 323
Interacting with the MASA Server 325
Interacting with MASA Through Web Application 327
Interacting with MASA Through REST APIs 330
Workflow to Provision a Router Using Ownership Voucher 331

Preface

This guide describes the configuration procedure and examples for system security in Cisco 8000 Series Routers.

- Changes to This Document, on page xiii
- Communications, Services, and Additional Information, on page xiii

Changes to This Document

This table lists the technical changes made to this document since it was first released.

Table 1: Changes to This Document

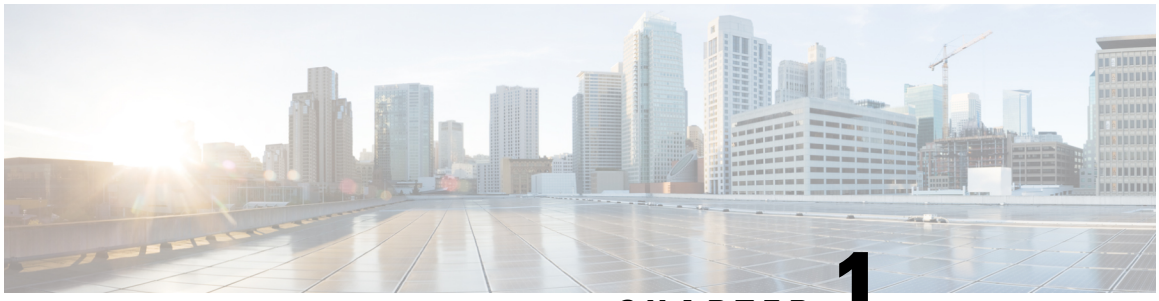
Date	Summary
March 2023	Initial release of this document

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

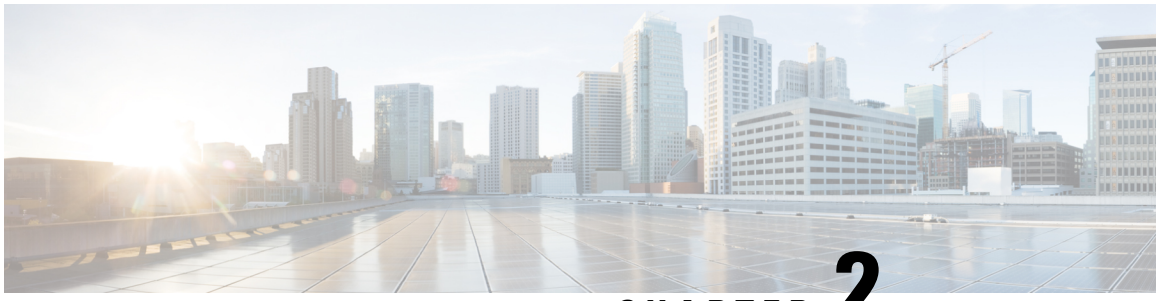
New and Changed Feature Information

This chapter lists all the features that have been added or modified in this guide. The table also contains references to these feature documentation sections.

- System Security Features Added or Modified in IOS XR Release 7.9.x, on page 1

System Security Features Added or Modified in IOS XR Release 7.9.x

Feature	Description	Changed in Release	Where Documented
Secure Key Integration Protocol (SKIP) for Routers	This feature was introduced.	Release 7.9.1	Secure Key Integration Protocol, on page 210
Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)	This feature was introduced.	Release 7.9.1	Configure X.509v3 Certificate-based Authentication for SSH, on page 281
uRPF in Strict Mode	This feature was introduced.	Release 7.9.1	Configuring uRPF in Strict Mode, on page 232



CHAPTER 2

YANG Data Models for System Security Features

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

• Using YANG Data Models, on page 3

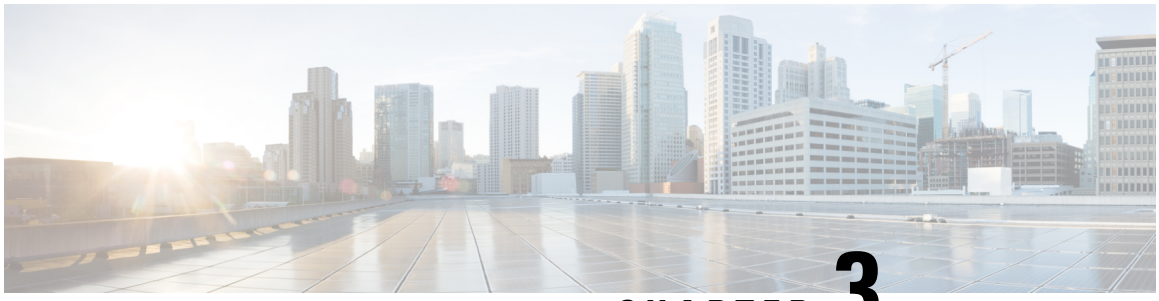
Using YANG Data Models

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

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

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

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



CHAPTER 3

Implementing Trustworthy Systems

This chapter describes the key components that form the trustworthy security system in Cisco 8000 Series Routers.

- Need for Trustworthy Systems, on page 5
- Enable Trust in Hardware, on page 6
- Enable Trust in Software, on page 9
- Establish and Maintain Trust at Steady State, on page 12
- How Trustworthiness is Implemented, on page 26
- Understanding Key Concepts in Security, on page 27

Need for Trustworthy Systems

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.

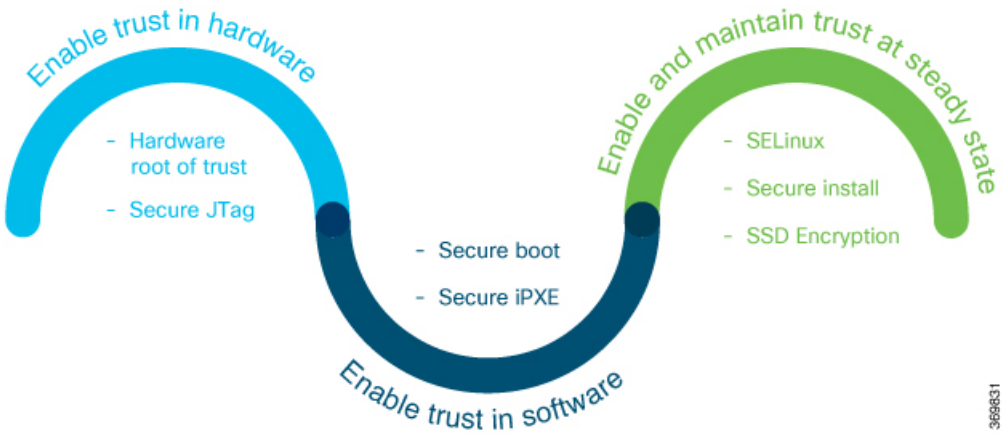
Routers are the critical components of the network infrastructure and must be able to protect the network and report on system integrity. A “trustworthy solution” is one that does what it is *expected* to do in a *verifiable* way. Building trustworthy solutions requires that security is a primary design 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 1: 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

Trust in the hardware in enabled through:

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 8000 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     hh4jzFBlxSGHZ4hKqnC2FEjqHg4tpx/chZ7YcTwLCco=
observed-digests:
Index   value
  0     hh4jzFBlxSGHZ4hKqnC2FEjqHg4tpx/chZ7YcTwLCco=
PCRs:
Index   value
  15    Dl1BGskyzeJ1LNYKuZK8Qqllwkth0ru+0xWydL9YMdc=
```

Hardware Integrity Check Using Chip Guard Functionality

The chip guard feature helps detect if attackers have replaced a Cisco router's Application Specific Integrated Circuit (ASIC) chip or CPU chip with a counterfeit one when the device is in the manufacturing supply chain. The ASIC performs critical functions, such as scanning an egress queue for error causes and a CPU runs the operating system. If these chips are counterfeited, the performance, reliability, and security of the router is compromised. During the hardware integrity check, at the time of device boot, if the chip guard feature identifies a counterfeit ASIC or CPU, it halts the secure boot process and displays a warning to inform that the supply chain integrity has been compromised.

Why do We Need Chip Guard

The increased hardware supply chain attacks compromise physical components, 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. Counterfeit chips in a router may have hidden functionalities to create a larger security vulnerability. Cisco's chip guard feature detects counterfeit chips before the router is deployed in the network.

Stages of Chip Guard Implementation

The table shows the various stages through which chip guard is implemented on the router.

Stage	Process/Action	Result
1. Router Manufacturing	SHA 256 hashes of the electronic chip IDs of both the CPU and ASIC are programmed in the TAm chip and stored in a database known as Imprint DB.	The Imprint DB inside the TAm chip contains the SHA 256 hashes, which cannot be modified during the router's lifetime.
2. Router Deployed in the Field and Powered Up	During the secure boot process, the chip guard feature recomputes the SHA 256 hashes of the electronic chip IDs of both the CPU and ASIC and creates a database known as Observed DB.	The Observed DB values are stored inside the TAm chip.
3. 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 notifies that either the CPU or ASIC is counterfeit, and the secure boot process halts. A message is displayed on the console about the chip guard validation failure.

Action to be Taken on Hardware Validation Failure

If you receive a chip guard warning about integrity check failure, you must create a service request on the Products Returns & Replacement (RMA) website.

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:
/1RDRAVACCkyXSBVFKZwNurif7DxQRvBBg6q9lhecKFZWOkp0FQACRVcAAAABXsfDQAA97/////AQAACQAAAAIAAAAQALAwEAAAAgrE798LlCkkplkcyTv50kG0/V46lIQutuSvgCUwjG8q4=
pcr-quote-signature:
mC8oPWYzgSIge31DLXCs/Ez7BRKsZyDVb4auuhJagWHa3aHSa9eMf34Y/FMuTitjeAhcs==<truncated>==dJYpsPKMGkcro1IquThaD1gKIH+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

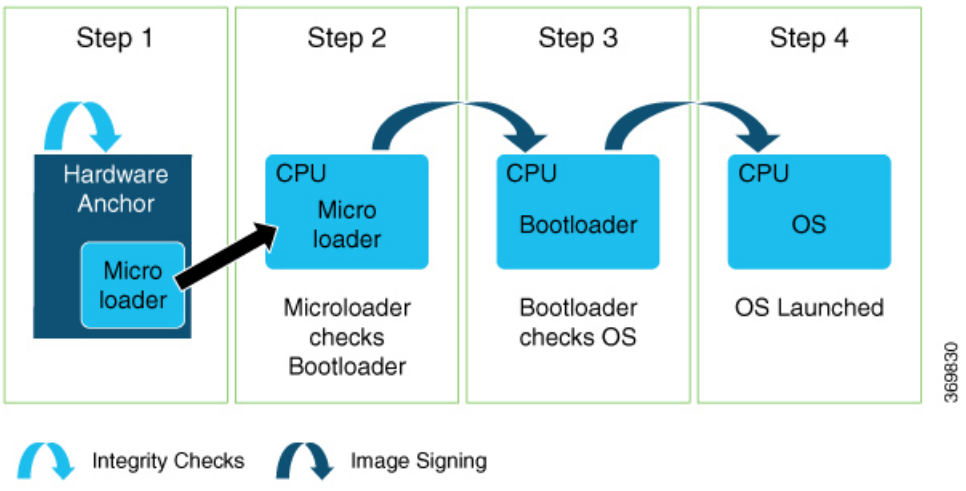
Secure Boot

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Secure Boot Status	Release 7.8.1	You can now verify whether the router is securely booted up with an authentic Cisco software image. We have introduced a **show** command to verify the secure boot status of the router. If the software image was tampered with, then the secure boot fails, and the router does not boot up. Before this release, there was no provision on the router to verify the secure boot status. The feature introduces these: • CLI: **show platform security integrity log secure-boot status** command. • YANG Data Model: `Cisco-IOS-XR-attestation-agent-oper.yang` Cisco native model (see GitHub)

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 2: 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.

Verify Secure Boot Status

Verify Secure Boot Status

Use the **show platform security integrity log secure-boot status** command to verify the secure boot status of the router. If the router boots up securely, then the **show** command output displays the status as *Enabled*. If the router does not support this secure boot verification functionality, then the status is displayed as *Not Supported*.

```
Router#show  platform security integrity log secure-boot status
Wed Aug 10 15:39:17.871 UTC

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

If the software image was tampered, then the secure boot fails and the router does not come up. The system displays corresponding error logs at various stages of boot up process. For example,

```
Bad signature file...
/initrd.img verification using Pkcs7 signature failed.
error: Security Violation: /initrd.img failed to load.
System halting...
```

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 3: Feature History Table

Feature Name	Release Information	Feature Description
Solid State Drive (SSD) Encryption	Release 7.3.1	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 *only* with a specific key stored in the TAm.

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:

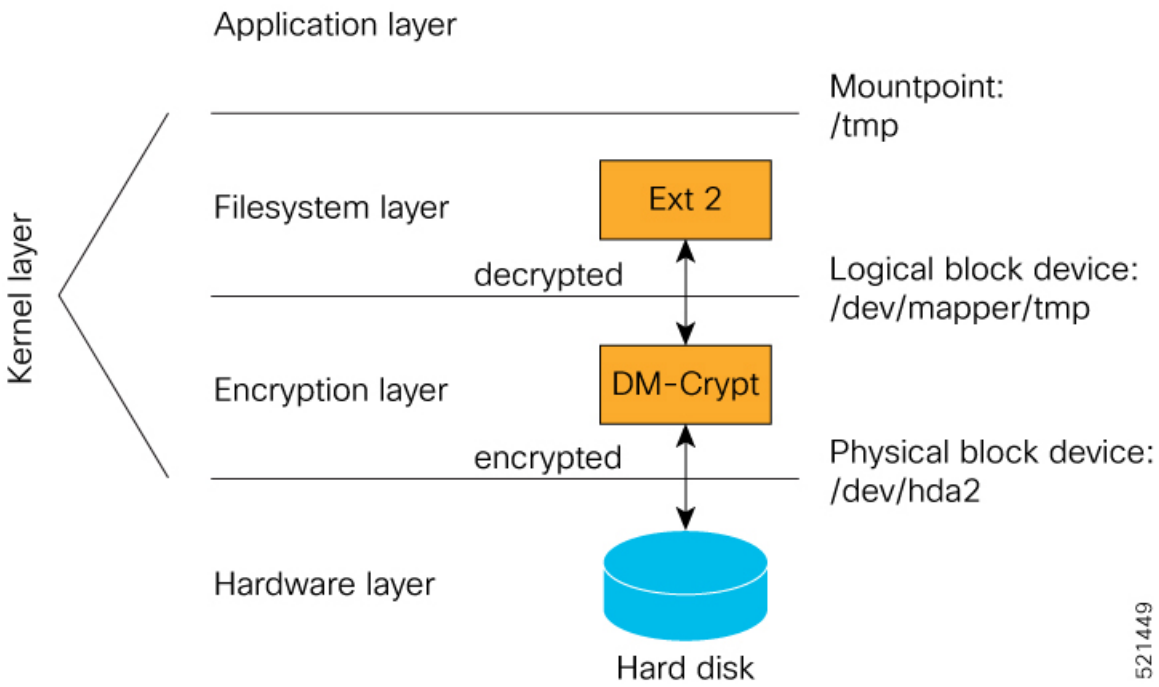
- DM-Crypt
- CPU with AES-NI support
- CryptSetup

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 3: DM-Crypt Encryption



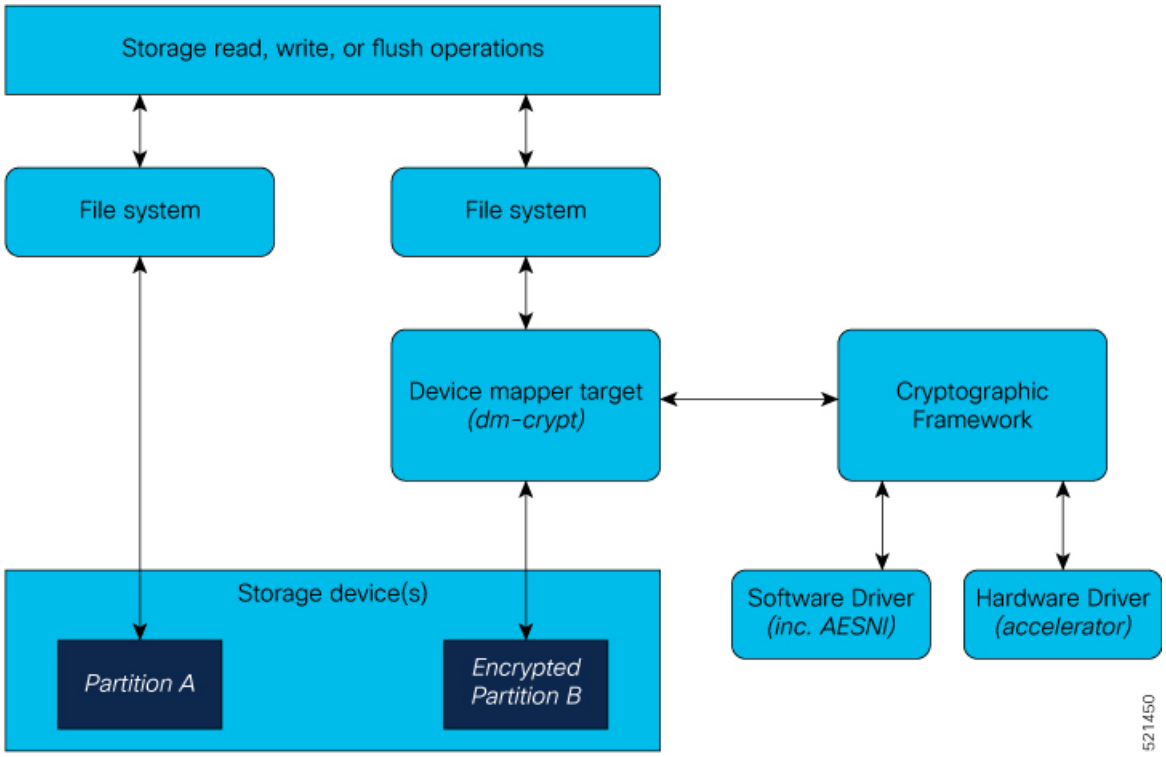
AES-NI Support

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 4: 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

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.

Runtime Defences (RTD)

Run Time Defenses (RTD) are a collection of tools used at runtime to mitigate common security vulnerabilities. RTD is classified into three groups:

Address Space Layout Randomization (ASLR)

ASLR is a technique that stops an attacker from getting access to a vulnerable program function, by preventing the attacker from finding out the memory address of the program function within the running process. To make this possible, ASLR randomly distributes the fundamental parts of the process (like the executable base, stack pointers, libraries, etc.) within the memory address space assigned to it by the operating system. Hence, attackers will never know the exact memory address of the program function and will be unable to exploit it. If they try to exploit by brute force, the process will crash.

Kernel Address Space Layout Randomization (KASLR)

Kernel Address Space Layout Randomization (KASLR) is a technique which is very similar to ASLR but it is applicable at the Linux kernel level. KASLR protects the Linux kernel by randomizing the fundamental parts of the kernel process within the kernel memory when the system boots. This stops an attacker from getting a pointer to the kernel memory distribution table in order to exploit it.

Built-in Object Size Checker (BOSC)

BOSC is a limited buffer overflow protection mechanism provided by the compiler. BOSC helps in avoiding buffer-overflow related security vulnerabilities on common SafeC library functions such as, strcpy_s and memcpy_s.

Executable Space Protection (XSpace)

XSpace mitigates malicious code injection attacks by protecting the data and code portions of the program memory. The basic approach of such attacks is to overflow a buffer in the program stack and cause the transfer of control to injected code. Once in the injected code, the application behaves in an unexpected manner.

X-space ensures that the area of the memory where the authentic code exists is write-protected and the area of memory where the data is present is execution-protected. Illegal execution-attempts in the data-portion or write-attempts in the code-portion results in the application crashing.

Note The XSpace functionality should not disabled at any time.

RTD Monitor

The RTD Monitor monitors all RTD functionalities on the router.

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 pcr 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/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hUwAAAEf/////
AQAAACQAAAALAAAAAQALAwCAAAAgae1J8QIYe06nS2RUx0JYeoG8tM3bqeVdpW7CObwBt+g=
pcr-quote-signature:
EZbzSUge89jSjH8ZqTgKJrZJBopEbd818C+h1Ec780qi7Li1WfCZQPIP6KCDV6HsRCVzLoFijgmlMLoZE2rakQq+/
```

```
1TgZOWSLjMY7RbjSFr8z/zbpVI+YLnOG+wytVYWuY33uKHBn/
YWokHwo+qVf7u9aLGhnrXKvRUaFknBiZtQGiyAdis6GbPTToqn0WSN1y6DPh4UHZj1vLVwJsI48mbQUrAyCZrz/
XBHLM38tVJjqSrC0jw/6LF2DDoT5ks0VUFT7sqbysw4F56y+z/IlDBrrRW3GFOY46MOxDxLwSl1/
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/SsvyNb8g3o7FFRGCZwfbs8EGxvMZg/PeN0NA71k=
observed-digests:
```

```
Index   value
  0     y3n/SsvyNb8g3o7FFRGCZwfbs8EGxvMZg/PeN0NA71k=
PCRs:
Index   value
  15    oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=
Cisco AIK Certificate used for signing PCR
pcr-quote: /1RDR4AYACCBG/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hywAAAEf////
/AQAAACQAAAALAAAAAQALAwCAAAAgae1J8QIYe06nS2RUx0JYeoG8tM3bqeVdpW7CObwBt+g=
pcr-quote-signature:
qyKbK7ndJbrgxeVnOodLWQzT7++NzrxJ9ERRvJzvTe4+8r6p0HGSepHUhZHzYkXw4DbniHAK0Cs3dwg/
hGKGe4M8Lz+/k682yIjaFYYip0DHMaV2ny/lT7RSqM/6u3j/JZrZv39MaeHa3MyjjonzRf9oe7EBSFAKsa/D54eTR0eFtaxFy/
XdtM0VVQe2JRdoBVxnIBLGiVmGRlVVlmHvwwgX1lAN6e3/soC1Vk3I5gjLldPHUYuJ/
7PTGyAwZsbdeigx8d4ViUUUjSMzK7JISwXa8k4GiPQVLBHtqqR+RA9scmMZTbKLsG3luIWKQeyCtXMYE1VOeW8WQlAvioMICw==
RP/0/RP0/CPU0:ios#show platform security integrity hardware digest-algorith$
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=QvZQd9q9psveoAz6QJQeNAAAAAAAAAAAAAAAAAAAAAA=
    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 liveliness 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 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 8000 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, third-party RPMs, ztp.ini, and secure ZTP certificates .

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)

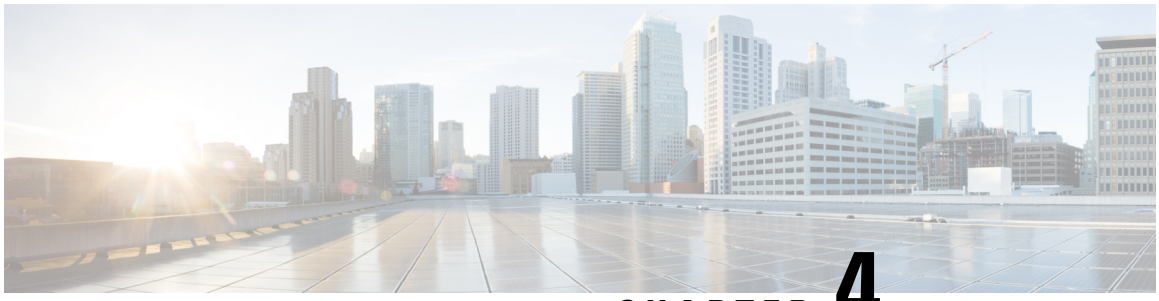
The Cisco Trust Anchor module (TAm) helps verify that Cisco hardware is authentic and provides additional security services.

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.



CHAPTER 4

Configuring AAA Services

This module describes the implementation of the administrative model of *task-based authorization* used to control user access in the Cisco IOS XR 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 Cisco IOS XR 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 Cisco IOS XR software base package and is available by default.

- Prerequisites for Configuring AAA Services, on page 31
- Restrictions for Configuring AAA Services, on page 31
- Information About Configuring AAA Services, on page 32
- How to Configure AAA Services, on page 50

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 45 section.

Information About Configuring AAA Services

This section lists all the conceptual information that a Cisco IOS XR 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 system user
- 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.

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. For more information, see the Task IDs for TACACS+ and RADIUS Authenticated Users, on page 45section.

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

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

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
- **maintenance:** Maintenance team tasks
- **netadmin:** Network administrator tasks
- **operator:** Operator day-to-day tasks (for demonstration purposes)
- **provisioning:** Provisioning team tasks
- **retrieve:** Retrieve team tasks
- **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. See the User Groups, on page 32 section.) 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.

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, Authorization Configuration, on page 86.
- 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. For more information, see the Task IDs for TACACS+ and RADIUS Authenticated Users, on page 45section.

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

Task Groups

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

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

AAA Configuration

This section provides information about AAA configuration.

Method Lists

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

Rollover Mechanism

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

The following methods are available:

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

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

Server Grouping

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

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

Authentication

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

Authentication of Root System User

The root-system user can log in to any node in any secure domain router in the system. A user is a root-system user if he or she belongs to the root-system group. The root-system user may be defined in the local or remote AAA database.

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.

Korn Shell Authentication

The korn shell (ksh) is the primary shell for the auxiliary port of the route processor (RP), standby RP, and distributed RP cards and for console and auxiliary ports of line cards (LCs) and service processors (SPs). The following are some of the characteristics of ksh authentication:

- For security reasons, ksh authentication allows only root-system users who have a secret configured. A root-system user with a normal password will not be authenticated because the normal password is two-way encrypted and poses a security risk because the password information is stored in the flash disk, which can be easily decrypted.
- Every time a root-system user with a secret is configured using the normal AAA CLI, that user is a valid ksh user and no separate configuration is required.
- Ksh does not authenticate TACACS+ or RADIUS users, even if they are root-system users.
- Ksh authentication uses a single user password database, which means when a root-system user on a dSC is configured using the normal AAA CLI, that user can log in using this username password in any card. This includes the RP, standby RP, LC, and SP.
- Ksh authentication cannot be turned off or bypassed after the card is booted. To bypass authentication, a user needs a reload of the card. (See the "Bypassing ksh Authentication" section for details).
- The ksh run from the console (using the **run** command) is not authenticated because the **run** command needs the root-system task ID. Because the user is already root-system, the user is not authenticated again.

Bypassing ksh Authentication

Although the authentication to ksh is lightweight and depends on very few processes, there are cases when ksh authentication needs to be bypassed, including the following:

- dSC (Active RP) disk0 corruption
- Loss of Qnet connectivity
- Inability to determine the node ID of the dSC (Active RP)

To bypass ksh authentication, the user has to set the ROMMON variable AUX_AUTHEN_LEVEL to 0 and then reload the image. A reboot is required only on the card that has to bypass authentication.

The ROMMON variable AUX_AUTHEN_LEVEL can have one of the following values:

- 0—Authentication will be bypassed on the card.
- 1—Loose authentication. Authentication is performed on a best-effort basis and permits the user to access ksh if the system cannot access authentication information successfully.
- 2—Strict authentication. This is the default state.

  Under no circumstances is authentication bypassed. Even if the authentication infrastructure is down, the system simply denies access.

For example, to bypass authentication on the card, enter the following:

```
rommon1> AUX_AUTHEN_LEVEL=0
rommon2> sync
rommon2> boot tftp:/ ...
```

Authentication Failure

In a system which is configured either with TACACS+ or RADIUS authentication with AAA configuration similar to the configuration below during the first login attempt or attempts, following a system reload, the login to the RP auxiliary port fails.

```
aaa authentication login default group tacacs+ group radius local
line template aux
login authentication default
```

This is because following the reload, the auxiliary port rejects login attempts with a valid TACACS+ configured *username* and *password*.

In such a scenario, the user has to first login with a valid locally configured *username* and *password*, and any login thereafter with TACACS+ configured *username* and *password*. Alternatively, if the user is connected to the auxiliary port via a terminal server, first clear the line used on the terminal server itself, and thereafter the user will be able to login to the auxiliary port with the TACACS+ configured *username* and *password*.

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 Encryption Methods

This feature provides the options for Type 8 and Type 9 encryption methods in AAA security services. The Type 8 and Type 9 encryption methods enable more secure and robust support for saving passwords with

respect to 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 encryption method uses SHA256 hashing algorithm, and the Type 9 encryption method uses scrypt hashing algorithm.

For more information about configuring users with Type 8 and Type 9 encryption methods, see Configure Users, on page 56 section.

Type 10 Password Encryption for User Management

The Cisco IOS XR 64 bit software supports Type 10 (**SHA512**) encryption algorithm for passwords used in user management. With this feature, **SHA512** is used by default for the passwords in user name configuration. This is applicable even for the first user creation. The **SHA512** encryption algorithm provides improved security to the user passwords compared to the older algorithms such as **MD5** and **SHA256**.

Restrictions for Type 10 Password Encryption Usage

The usage of Type 10 password encryption is subjected to this restriction:

- In a first user configuration scenario or when a user is reconfigured, only the Type 5 and Type 10 encryption are synced from XR VM to System Admin VM and Host VM; Type 8 and Type 9 are not synced.

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 61.

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:

```
Router: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:

```
Router: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:

```
Router: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 54.

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 63.

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

Table 4: 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:

```
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.

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:
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:
Router# show user tasks
No task ids available
```

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:
  - AppleTalk Remote Access (ARA)
  - 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.

Differentiated Services Code Point (DSCP) Marking Support for TACACS Packets

Differentiated Services is a Quality of Service (QoS) architecture that manages the data traffic in a network by using the principle of traffic classification. In this model, the traffic is divided into classes and the data packets are forwarded to the corresponding classes. Based on the priority of the network traffic, the different classes are managed.

To classify traffic, Differentiated Services uses Differentiated Services Code Point (DSCP). It is a 6-bit field in the Type of Service (ToS) byte in the IP header. Based on the DSCP value, the user is able to classify the data traffic and forward packets to the next destination.

You can set the value of DSCP. For a single connection, set the DSCP value on the socket while connecting to the server. In this way, all the outgoing packets will have the same DSCP value in their IP headers. For multiple connections, the DSCP value is set on the available open sockets. Use the **tacacs-server ipv4** command to set the DSCP value.

How to Configure AAA Services

To configure AAA services, perform the tasks described in the following sections.

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:

```
Router(config)# taskgroup beta
```

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

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

Step 3 **description** *string*

Example:

```
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:

```
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.

Task Group Configuration

Task groups are configured with a set of task IDs per action type.

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:

```
Router(config)# taskgroup beta
```

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

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

Step 3 **description** *string*

Example:

```
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:

```
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 Step 4 for each task ID to be associated with the task group named in Step 2.

—

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

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

end —Prompts user to take one of these actions:

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

What to do next

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

Configure User Groups

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

Before you begin

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
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:

```
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:

```
Router(config-ug)#
inherit usergroup sales
```

- Explicitly defines permissions for the user group.

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

Example:

```
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 43.

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.

Note You must not use the following words as usernames:

- backup
- bin
- bind
- daemon
- dhcp
- games
- gnat
- irc
- ip
- list
- mail
- man
- messagebus
- news
- nobody
- proxy
- rpc
- root
- sys
- sync
- systemd-timesync
- systemd-network
- systemd-bus-proxy
- sshd
- uucp

- www-data

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
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**|**8** | **9**| **10**} *secret*

Example:

```
Router(config-un)# password 0 pwd1
```

or

```
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** 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:

```
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 5: Feature History Table

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

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

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

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

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

Enter password:
Re-enter password:

Router(config)#commit
```

View the encrypted password:

```
Router# show run aaa
..

username us3
 password 7 105A1D0D
```

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

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

```
Enter password:
Re-enter password:


Router(config)#commit
```

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

Configure Type 8 and Type 9 Passwords

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

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

The Type 5, Type 8, Type 9 and Type 10 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 56 section.

Configuration Example

Directly configuring a Type 8 encrypted password:

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

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

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

Directly configuring a Type 9 encrypted password:

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

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

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

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

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

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

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

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

Enter secret:
```

```
Re-enter secret:

Router(config)# commit
```

View the encrypted secret:

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

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

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

Enter secret:
Re-enter secret:

Router(config)# commmit
```

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

Related Topics

- Type 8 and Type 9 Encryption Methods, on page 39
- Type 10 Password Encryption for User Management, on page 40

Associated Commands

- secret
- username

Configure Type 10 Password Encryption

You can use these options to configure Type 10 (**sha512**) password encryption for the user:

Configuration Example

The Type 10 encryption is applied by default 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 explicitly configure Type 10 encryption.

```
Router#configure
Router(config)#username root secret 10 $6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
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
```

```
Router#show run aaa
!
username user10
secret 10 $6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
!
```

The above configuration returns the encrypted password using Type10 algorithm (use the **show run username** command to verify that) which can then be configured for the user as follows:

```
Router(config)#username user10 secret 10 $6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
Router(config-un)#commit
```

Running Configuration

```
Router#show run username user10
!
username user10
secret 10 $6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
!
```

Related Topics

- Type 10 Password Encryption for User Management, on page 40

Associated Commands

- username
- secret

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.

```
Router(config)#aaa password-policy test-policy
Router(config-aaa)#min-length 8
Router(config-aaa)#max-length 15
Router(config-aaa)#lifetime months 3
Router(config-aaa)#min-char-change 5
Router(config-aaa)#authen-max-attempts 3
Router(config-aaa)#lockout-time days 1
Router(config-aaa)#commit
```

```
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:

```
Router#show aaa password-policy

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 40

Associated Commands

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

Configure Password Policy for User Secret and Password

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

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

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

Guidelines to Configure Password Policy for User Secret

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

- If there is no policy already configured while configuring the user secret, then the system does not have any policy validation to do for that secret. So, you must ensure that the policy is configured first and then applied to the username configuration, before configuring the secret. Especially when you copy and paste the username configurations.
- If you change the user secret at the time of log in, the system applies the same hashing type as it was applied in the username configuration. For example, if the secret was applied as Type 5 in the username configuration, then the system applies Type 5 itself if the secret is modified at the time of log in.
- Password and secret are different entities. Hence, if **restrict-old-count** is configured in the policy while changing the password, the system checks for compliance only with the history of old passwords; not with the history of old secrets.
- Similarly, the system does not check for old password history while changing the secret and vice versa. So, if the same secret (in clear text) was used before as password for the user, then the system allows that secret configuration. And, vice versa, for the password configuration.
- The **restrict-old-count** applies to both secret and password. So, the configured secret or password overwrites the old secret or password in the FIFO order.
- When you try to assign a different policy to a username which already has a password or secret associated to a policy, then the system rejects that configuration. The error message indicates to remove the existing password or secret in order to apply the new policy to the user.
- The system does not allow any configuration that requires the secret to be validated against the previous composition of the cleartext secret. This is because, you cannot retrieve the clear text format of the secret that was once hashed, for comparison. Hence, the following configurations do not have any effect on the secret configuration of the user:
  - **max-char-repetition**
  - **min-char-change**
  - **restrict-password-reverse**
  - **restrict-password-advanced**
- As the new **policy** configuration for the user is common to password and secret, the existing **password-policy** configuration becomes redundant. So, these configurations must be mutually exclusive. When any one of these configurations is already present, and if you try to configure the other policy, then the system rejects it. The error message says that **password-policy** and **policy** are not allowed together.

Configuration Example

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

```
Router#configure
Router(config)#username user1
Router(config-un)#policy test-policy1
Router(config-un)#secret 10
```

```
$6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZLhoHd7TicR4mOo8IIVriYCGAKW0A.w1JvTPO7IbZry.DxHrE3SN2BBzBJe0
Router(config-un)#commit
```

Running Configuration

```
username user1
policy test-policy1
secret 10
$6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZLhoHd7TicR4mOo8IIVriYCGAKW0A.w1JvTPO7IbZry.DxHrE3SN2BBzBJe0
!
```

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

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

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

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

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

```
(1)aaa password-policy pol1
```

```
 lifetime minutes 1
 upper-case 1
 restrict-old-count 2
!

username lab2
 group root-lr
(2) policy pol1
(3) secret 10
$6$gphqA0RfBXOn6A0.$wRwWG1l0TIpHPdVQ66fUiIM5P46ggoGMGgFuaZd0LD2DLFYDlDPaRyXQLi8Izjb49tC7H7tkTLrc1.GELFpiK.

  password 7 1533292F200F2D
!
```

Related Topics

- Password Policy for User Secret, on page 43

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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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 Per VRF AAA

The Per VRF AAA functionality enables AAA services to be based on VPN routing and forwarding (VRF) instances. The Provider Edge (PE) or Virtual Home Gateway (VHG) communicates directly with the customer's RADIUS server, which is associated with the customer's VPN, without having to go through a RADIUS proxy. Thus, ISPs can scale their VPN offerings more efficiently, because they no longer have to use RADIUS proxies and they can provide their customers with the flexibility they demand.

New Vendor-Specific Attributes (VSAs)

The Internet Engineering Task Force (IETF) draft standard specifies a method for communicating vendor-specific information between the network access server and the RADIUS server by using the vendor-specific attribute (attribute 26). Attribute 26 encapsulates vendor-specific attributes, thereby, allowing vendors to support their own extended attributes otherwise not suitable for general use.

The Cisco IOS XR software RADIUS implementation supports one vendor-specific option using the format recommended in the specification. Cisco's vendor-ID is 9, and the supported option has vendor-type 1, which is named “cisco-avpair ” The value is a string of the following format:

```
protocol : attribute sep value *
```

“Protocol” is a value of the Cisco “protocol ” attribute for a particular type of authorization. “Attribute” and “value” are an appropriate attribute-value (AV) pair defined in the Cisco RADIUS specification, and “sep” is “=” for mandatory attributes and “*” for optional attributes.

This table describes the VSAs that are now supported for Per VRF AAA.

Table 6: Supported VSAs for Per VRF AAA

VSA Name	Value Type	Description
Note The RADIUS VSAs—rad-serv, rad-serv-source-if, and rad-serv-vrf—must have the prefix “aaa:” before the VSA name.		

VSA Name	Value Type	Description

rad-serv	string	Indicates the IP address in IPv4 or IPv6 format, key, timeout, and retransmit number of a server and the group of the server.
		The VSA syntax follows:
		`rad-serv=a.b.c.d [key SomeKey] [auth-port X] [acct-port Y]`
		`[retransmit V] [timeout W].`
		Other than the IP address, all parameters are optional and are issued in any order. If the optional parameters are not specified, their default values are used.
		The key cannot contain any spaces; for "retransmit V," "V" can range from 1 to 100; for "timeout W," the "W" can range from 1 to 1000.

rad-serv-vrf	string	Specifies the name of the VRF that is used to transmit RADIUS packets. The VRF name matches the name that was specified through the **vrf** command.

This task configures RADIUS server groups per VRF. For information about configuring TACACS+ server groups per VRF, refer Configure TACACS+ Server Groups, on page 80.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# aaa group server radius radgroup1
Router(config-sg-radius)#
```

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

Step 3 **server-private** {*hostname* | *ip-address in IPv4 or IPv6 format*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*]

Example:

IP address in IPv4 format

```
 Router(config-sg-radius)# server-private 10.1.1.1 timeout 5
Router(config-sg-radius)# server-private 10.2.2.2 retransmit 3
```

Example:

IP address in IPv6 format

```
 Router(config-sg-radius)# server-private 2001:db8:a0b:12f0::1/64 timeout 5
Router(config-sg-radius)# server-private 10.2.2.2 retransmit 3
```

Configures the IP address of the private RADIUS server for the group.

If private server parameters are not specified, global configurations are used. If global configurations are not specified, default values are used.

Both **auth-port** and **acct-port** keywords enter RADIUS server-group private configuration mode.

You can configure a maximum of 30 private servers per RADIUS server group.

Step 4 **vrf** *vrf-name*

Example:

```
 Router(config-sg-radius)# vrf v2.44.com
```

Configures the VRF reference of an AAA RADIUS server group.

Note
Private server IP addresses can overlap with those configured globally and the VRF definitions can help to distinguish them.

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 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:

```
Router(config)# tacacs-server host 209.165.200.226 port 51
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:

```
Router(config-tacacs-host)# tacacs-server host 209.165.200.226 timeout 30
```

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:

```
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:

```
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 7.3.2)
- 5 to 7200 (from Cisco IOS XR Software Release 7.3.2, and later)

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

Example:

```
Router(config)# tacacs source-interface GigabitEthernet 0/4/0/0 vrf abc
```

(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:

```
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 Authorization for a TACACS+ Server

This task helps you configure authorization commands are used to verify that an authenticated user (or principal) is granted permission to perform a specific task.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **aaa authorization command group tacacs | none**

Example:

```
Router(config)# aaa authorization command group tacacs
```

Configure the AAA system to perform remote authorization using TACACS+ protocol.

Example:

```
Router(config)# aaa authorization command group none
```

Configure the AAA system to not perform any authorization.

Example:

```
Router(config)# aaa authorization command group tacacs none
```

Configure the AAA system to first perform TACACS+ authorization and if it fails, no authorization should be performed.

Step 3 **confdConfig aaa authorization enabled**

Example:

```
Router(config)# confdConfig aaa authorization enabled
```

Configure ConfD to perform remote authorization.

Step 4 **confdConfig aaa authorization callback enabled**

Example:

```
Router(config)# confdConfig aaa authorization callback enabled
```

Configure ConfD to invoke application callbacks for authorization.

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 Authentication for a TACACS+ Server

This task describes how to configure authentication commands to verify the identity of a user or principal TACACS+server.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **confdConfig aaa externalAuthentication enabled**

Example:

```
Router(config)# confdConfig aaa externalAuthentication enabled
```

Configure Confd to perform external authentication.

Step 3 **confdConfig aaa authOrder** localAuthentication|externalAuthentication

Example:

```
Router(config)# confdConfig aaa authOrder externalAuthentication localAuthentication
```

Configure the AAA subsytem to perform external authentication first and then local authentication.

Step 4 **confdConfig aaa externalAuthentication executable**"chvrf 0 /opt/cisco/calvados/bin/calvados_login_aaa_proxy"

Example:

```
Router(config)# confdConfig aaa externalAuthentication executable chvrf 0
/opt/cisco/calvados/bin/calvados_login_aaa_proxy
```

Configure the AAA system to perform external authentication using login executable configured on local host.

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 Accounting for a TACACS+ Server

This task describes how to configure accounting commands that are used for logging of sessions and to create an audit trail by recording certain user- or system-generated actions.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **aaa accounting command tacacs**

Example:

```
Router(config)# aaa accounting command tacacs
```

Configure remote accounting commands.

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.

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:

```
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:

```
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:

```
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:

```
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).

You can configure a maximum of :

- 10 TACACS+ servers per server group
- 10 private TACACS+ servers

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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
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:

```
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 (Optional) **vrf** *vrf-id*

Example:

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

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

Step 5 Repeat step 3 for every external server to be added to the server group named in step 2.

—

Step 6 (Optional) vrf *vrf-id*

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:

```
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**

Configure AAA 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.

This section contains the following procedures:

Configuring Authentication Method Lists

This task configures method lists for authentication.

Authentication Configuration

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.

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:

```
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.

—

Configuring Authorization Method Lists

This task configures method lists for authorization.

Note You can configure the **radius** keyword for the **aaa authorization** command.

Authorization Configuration

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.

Method lists are specific to the type of authorization being requested. Four types of AAA authorization are supported:

- **Commands authorization—**Applies to the XR EXEC mode mode commands a user issues. Command authorization attempts authorization for all XR EXEC mode mode commands.

Note "Command" authorization is distinct from "task-based" authorization, which is based on the task profile established during authentication.

- **XR EXEC mode authorization—**Applies authorization for starting XR EXEC mode session.
- **Network authorization—**Applies authorization for network services, such as IKE.

- **Eventmanager authorization—**Applies an authorization method for authorizing an event manager (fault manager). RADIUS servers are not allowed to be configured for the event manager (fault manager) authorization. You are allowed to use TACACS+ or locald.

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** 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**.

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.

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:

```
Router(config)# aaa authorization commands listname1 group tacacs+
```

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

- The **commands** keyword configures authorization for all XR EXEC mode shell commands. Command authorization applies to the EXEC mode commands issued by a user. Command authorization attempts authorization for all XR EXEC modecommands.
- The **eventmanager** keyword applies an authorization method for authorizing an event manager (fault manager).
- The **exec** keyword configures authorization for an interactive (XR EXEC mode) session.
- The **network** keyword configures authorization for network services like PPP or IKE.
- The **default** keyword causes the listed authorization methods that follow this keyword to be the default list of methods for authorization.

- A *list-name* character string identifies the authorization method list. The method list itself follows the method list name. Method list types are entered in the preferred sequence. The listed method list types can be any one of the following:
  - **none**—The network access server (NAS) does not request authorization information. Authorization always succeeds. No subsequent authorization methods will be attempted. However, the task ID authorization is always required and cannot be disabled.
  - **local**—Uses local database for authorization.
- **group tacacs+**—Uses the list of all configured TACACS+ servers for authorization. The NAS exchanges authorization information with the TACACS+ security daemon. TACACS+ authorization defines specific rights for users by associating AV pairs, which are stored in a database on the TACACS+ security server, with the appropriate user.
- **group radius**—Uses the list of all configured RADIUS servers for authorization.
- **group** *group-name*—Uses a named server group, a subset of TACACS+ or RADIUS servers for authorization as defined by the **aaa group server tacacs+** or **aaa group server radius** command.

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

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

end —Prompts user to take one of these actions:

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

Configuring Accounting Method Lists

This task configures method lists for accounting.

Note You can configure the **radius** keyword for the **aaa accounting** command.

Accounting Configuration

Currently, Cisco IOS XR 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.

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:

```
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:

```
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.

Applying Method Lists for Applications

After you configure method lists for authorization and accounting services, you can apply those method lists for applications that use those services (console, vty, and so on). Applying method lists is accomplished by enabling AAA authorization and accounting.

This section contains the following procedures:

Enabling AAA Authorization

This task enables AAA authorization for a specific line or group of lines.

Method List Application

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** {**aux** | **console** | **default** | **template** *template-name*}

Example:

```
Router(config)# line console
```

Enters line template configuration mode.

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

Example:

```
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. (See the Enable Accounting Services, on page 94section.)

Enable Accounting Services

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# line console
```

Enters line template configuration mode.

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

Example:

```
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:

```
Router(config)# line template alpha
```

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

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

Example:

```
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.

Command Accounting

Command accounting with a method as local, enables the logging of commands executed by all users as syslog messages. This feature can be enabled or disabled only by users who have AAA write permissions. Once enabled, all the commands that are executed by all users can be viewed from the output of the **show logging** command.

Command accounting is not supported for commands that are executed using Netconf, XML or GRPC. Command accounting is not used as a failover accounting method but as an additional method of accounting. So this feature will be active even when other accounting methods are configured and functional.

Configuring Command Accounting

Command Accounting can either be configured alone or along with other accounting methods as shown below:

1. Configuring command accounting alone

```
Router(config)# aaa accounting commands default start-stop local none
Router(config)# commit
```

2. Configuring command accounting along with other accounting methods

```
Router(config)# aaa accounting commands default start-stop group tacacs+ local none
Router(config)# commit
```

Command Authorization Using Local User Account

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Command Authorization Using Local User Account	Release 7.5.1	This feature allows locally authenticated users—authenticated by the AAA server internal to the router—to run all XR VM commands even if a remote TACACS+ AAA server is not reachable for authorization. It prevents a complete router lockdown. The feature also prevents remotely authenticated users—authenticated using a remote AAA server (say, TACACS+ server)—from running any non-permitted commands on the router, and thus prevents misuse of user privileges. This feature modifies the aaa authorization commands default command to include the **local** option for XR VM command authorization.

Currently, when a user tries to execute a command on XR VM, the router checks to see whether the user has required permissions to execute it. The router does this authorization process in two steps. First, the system compares the task-IDs of the user with the required task-IDs for the command. If the user has all required task-IDs, and if AAA authorization is configured, then the system sends an authorization request to the local or remote AAA server, based on that configuration. Based on the response from the AAA server, the system allows or rejects the command execution. If authorization is not configured or if it configured with option *none*, then the system bypasses authorization check and allows user to execute the command.

Similarly, the existing remote authorization process using TACACS+ server has two options—remote authorization using *tacacs+* and *none*. The authorization process using *TACACS+* option uses an external TACACS+ server for authorization. The authorization using *none* option allows the user to execute the command without any authorization check. TACACS+ authorization has the advantage of fine-tuning authorization rules and providing more control on system access that cannot be otherwise done locally. However, if the remote server is not reachable, a user who leverages TACACS+ authorization might get into an unpredictable state of router, as mentioned in these scenarios:

- Remote authorization using *TACACS+* with failover option as *none* (that is, with the **aaa authorization commands default group tacacs+ none** configuration)

  If TACACS+ server is not reachable, then the system bypasses the authorization check and allows user to execute the command. A user who does not have permission to execute certain commands due to additional authorization rules on the TACACS+ server, then gets permission to execute those commands in this scenario. This action introduces a privilege escalation.

- Remote authorization using *TACACS+* without any failover option (that is, with the **aaa authorization commands default group tacacs+** configuration)

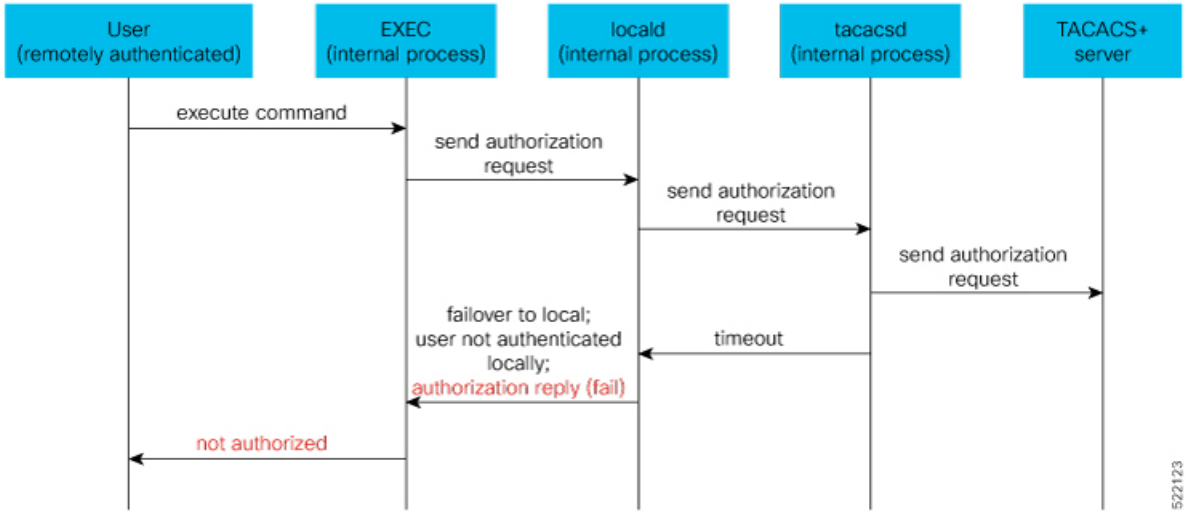
  If TACACS+ server is not reachable, then the system does not authorize the command at all. Because the user then cannot execute any command, the router gets locked out.

With the introduction of command authorization using local user account feature in Cisco IOS XR Software Release 7.5.1, locally authenticated users can execute commands even if a TACACS+ server is not reachable. This behavior is similar to the behavior with the failover option *none*, with the only difference that only locally authenticated users can execute commands in this case. This functionality thereby prevents a complete lockdown of the router as mentioned in one of the previously existing scenarios mentioned earlier. At the same time, the feature also prevents users who are authenticated remotely (that is, TACACS+ authenticated users) from executing any non-permitted command on the router. This behavior in turn helps to prevent any sort of misuse of user privileges on the router.

Call Flow of Command Authorization

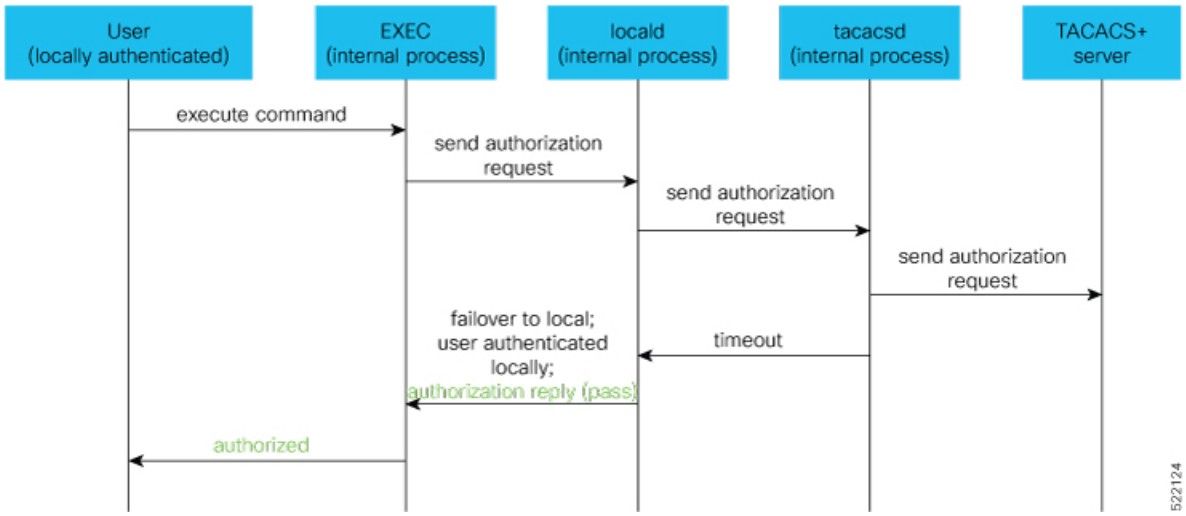
Consider a scenario where the user is remotely authenticated. In the event of timeout from the TACACS+ server, the command authorization fails. The user cannot execute any command until the TACACS+ server is reachable again, thereby preventing misuse of user privileges on the router.

Figure 5: Call Flow of Command Authorization for Remotely Authenticated Users



Consider a scenario where the user is locally authenticated. The command authorization still succeeds even if the authorization request to the TACACS+ server times out. There is no additional check done by the local AAA component in the router. As a result, the user can execute the command irrespective of the fact that the TACACS+ server is not reachable. This functionality prevents a complete lockdown of the router.

Figure 6: Call Flow of Command Authorization for Locally Authenticated Users



Configure Command Authorization Using Local User Account

Guidelines

Although there is no restriction in configuring local command authorization, you must be cautious to prevent any potential lockout due to misconfiguration. For instance, if *local* is the only method of authorization specified for the commands, a remotely authenticated user configuring command authorization using local user account feature cannot execute further commands.

Configuration Example

To configure command authorization using local user account, use the **local** option in the **aaa authorization** command in any of these formats:

```
Router#configure
Router(config)#aaa authorization commands default group tacacs+ local
```

Or

```
Router(config)#aaa authorization commands default local
```

Running Configuration

```
Router#show run aaa
!
aaa authorization commands default group tacacs+ local
!
```

```
Router#show run aaa
!
aaa authorization commands default local
!
```

Verification

```
Router#show user authentication method
local
```

Feature Behavior and Use Case Scenarios

Feature Behavior With Various Local Command Authorization Options

This table lists the feature behavior scenarios with various local command authorization options.

Table 8: Feature Behavior with Various Local Command Authorization Options

AAA Configuration	Expected Behavior
aaa authorization commands default group tacacs+ local	If TACACS+ server is not reachable, system allows locally authenticated users to execute the command. If TACACS+ server is reachable and if it returns an authorization failure, then the system does not perform any failover to local authentication with this configuration.
aaa authorization commands default local	This configuration allows only locally authenticated users to execute commands. System completely blocks remote users from executing any command.
aaa authorization commands default local group tacacs+	In this scenario, system chooses local authorization first and grants access if the user is locally authenticated. If not, the request fails over to TACACS+ server. This combination of command options is useful when both local and remote authenticated users want to execute commands when TACACS+ server is reachable.
aaa authorization commands default local none	Although configurable, this combination of command options does not provide any additional security with respect to user access. It is equivalent to having no authorization.

Use Case Scenarios of Command Authorization

In the following scenarios, local user refers to user whose is authenticated locally and whose profile is available locally, but not available on the remote server (TACACS+ server). Similarly, remote user refers to user whose is authenticated remotely and whose profile is available on the remote server, but not available locally. And, both local user and remote user are considered to have *root-lr* permission to execute the commands, in these scenarios.

Table 9: Use Case Scenarios of Command Authorization

Type of User (local or remote)	AAA Configuration Summary	Use Case Scenario	Expected Behavior
Local and remote user	No command authorization configured	Execute a command	Command authorization succeeds if the required task-IDs are available
Local user	Only *tacacs+ command authorization* configured.	Execute a command when TACACS+ server is reachable	Command authorization fails
		Execute a command when TACACS+ server is not reachable	Command authorization fails

Remote user	Only *tacacs+ command authorization* configured	Execute a command when TACACS+ server is reachable	Command authorization succeeds
			`Router#show run aaa authorization`
			`aaa authorization commands default group tacacs+`

		Execute a command when TACACS+ server is not reachable	Command authorization fails
Local user	Only *tacacs+ command authorization* configured with failover option as *none*.	Execute a command when TACACS+ server is reachable	Command authorization fails

		Execute a command when TACACS+ server is not reachable	Command authorization succeeds
			`Router#show user authentication method`
			`local`

Remote user	Only *tacacs+ command authorization* configured with failover option as *none*.	Execute a command that is restricted only to that user when TACACS+ server is reachable	Command authorization fails
		Execute a command that is restricted only to that user when TACACS+ server is not reachable	Command authorization succeeds

Type of User (local or remote)	AAA Configuration Summary	Use Case Scenario	Expected Behavior
Local user	Only *local command authorization* configured.	Execute a command	Command authorization succeeds `Router#`**`show run aaa authentication`** `aaa authentication login default group tacacs+ local`
Remote user	Only *local command authorization* configured.	Execute a command	Command authorization fails
Local user	Only *tacacs+ command authorization* configured with failover option as *local*.	Execute a command when TACACS+ server is reachable	Command authorization fails
		Execute a command when TACACS+ server is not reachable	Command authorization succeeds `Router#`**`show run aaa authorization`** `aaa authorization commands default group tacacs+ local`
Remote user	Only *tacacs+ command authorization* configured with failover option as *local*.	Execute a command when TACACS+ server is reachable	Command authorization succeeds `Router#`**`show run aaa authorization`** `aaa authorization commands default group tacacs+ local`
		Execute a command when TACACS+ server is not reachable	Command authorization fails

Configuration Example for AAA Services

The following examples show how to configure AAA services.

An authentication method list vty-authen is configured. This example specifies a method list that uses the list of all configured TACACS+ servers for authentication. If that method fails, the local username database method is used for authentication.

```
configure
aaa authentication login vty-authen group tacacs+ local
```

The default method list for PPP is configured to use local method.

```
aaa authentication ppp default local
```

A username user1 is created for login purposes, a secure login password is assigned, and user1 is made a root-lr user. Configure similar settings for username user2.

```
username user1
secret lab
group root-lr
exit

username user2
secret lab
exit
```

A task group named tga is created, tasks are added to tga, a user group named uga is created, and uga is configured to inherit permissions from task group tga. A description is added to task group uga.

```
taskgroup tga
task read bgp
task write ospf
exit

usergroup uga
taskgroup tga
description usergroup uga
exit
```

Username user2 is configured to inherit from user group uga.

```
username user2
group uga
exit
```

Three TACACS servers are configured.

```
tacacs-server host 10.1.1.1 port 1 key abc
tacacs-server host 10.2.2.2 port 2 key def
tacacs-server host 10.3.3.3 port 3 key ghi
```

A user group named priv5 is created, which will be used for users authenticated using the TACACS+ method and whose entry in the external TACACS+ daemon configuration file has a privilege level of 5.

```
usergroup priv5
taskgroup operator
exit
```

An authorization method list, vty-author, is configured. This example specifies that command authorization be done using the list of all configured TACACS+ servers.

```
aaa authorization commands vty-author group tacacs+
```

An accounting method list, vty-acct, is configured. This example specifies that start-stop command accounting be done using the list of all configured TACACS+ servers.

```
aaa accounting commands vty-acct start-stop group tacacs+
```

For TACACS+ authentication, if, for example, a privilege level 8 is returned, and no local usergroup priv8 exists and no local user with the same name exists, the **aaa default-taskgroup** command with tga specified as the *taskgroup-name* argument ensures that such users are given the taskmap of the task group tga.

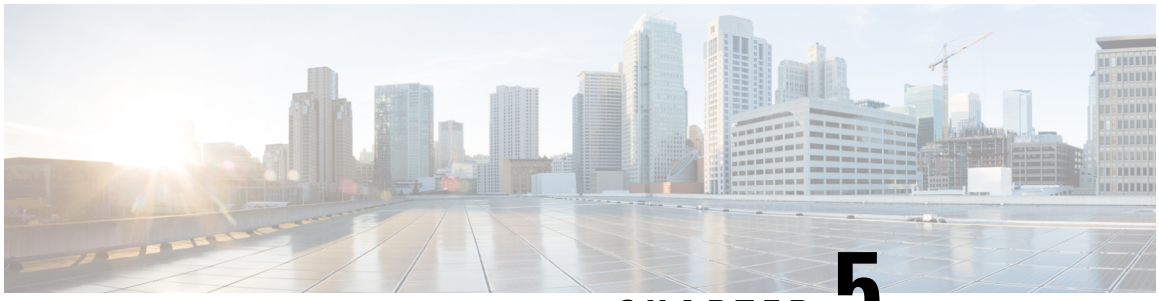
```
aaa default-taskgroup tga
```

For line template vty, a line password is assigned that is used with line authentication and makes usergroup uga the group that is assigned for line authentication (if used), and makes vty-authen, vty-author, and vty-acct, respectively, the method lists that are used for authentication, authorization, and accounting.

```
line template vty
password lab
users group uga
login authentication vty-authen
authorization commands vty-author
accounting commands vty-acct
exit
```

A TACACS+ server group named abc is created and an already configured TACACS+ server is added to it.

```
aaa group server tacacs+ abc
server 10.3.3.3
exit
```



CHAPTER 5

Implementing Certification Authority Interoperability

Certification authority (CA) interoperability is provided in support of the IP Security (IPSec), Secure Socket Layer (SSL), and Secure Shell (SSH) protocols. This module describes how to implement CA 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 is not currently supported.

Feature History for Implementing Certification Authority Interoperability

Release	Modification
Release 7.0.12	This chapter was introduced.
Release 7.3.1	Added support for verifying authenticity of RPM packages using runtime and install time fingerprint.
Release 7.3.1	Added support to collect filesystem inventory.
Release 7.3.1	Added support for new optimizations via IMA.
Release 7.3.1	Added support for retrieving CRL through the http proxy server.

- Implementing Certification Authority Interoperability, on page 105
- Information About Implementing Certification Authority, on page 147

Implementing Certification Authority Interoperability

Certification authority (CA) interoperability is provided in support of the IP Security (IPSec), Secure Socket Layer (SSL), and Secure Shell (SSH) protocols. This module describes how to implement CA 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 is not currently supported.

Feature History for Implementing Certification Authority Interoperability

Release	Modification
Release 7.0.12	This chapter was introduced.
Release 7.3.1	Added support for verifying authenticity of RPM packages using runtime and install time fingerprint.
Release 7.3.1	Added support to collect filesystem inventory.
Release 7.3.1	Added support for new optimizations via IMA.
Release 7.3.1	Added support for retrieving CRL through the http proxy server.

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 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]).

How to Implement CA Interoperability

This section contains the following procedures:

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:

```
Router(config)# hostname myhost
```

Configures the hostname of the router.

Step 3 **domain name domain-name**

Example:

```
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

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
RSA and DSA Keys Available in Running Configuration	Release 7.3.4	You can now view the RSA and DSA keys in the running configuration by using the **show running-configuration** command. This feature is applicable to the following sections: • Generate Key Pair, on page 130 • Generate Key Pair, on page 130 • Configure FIPS-compliant Keys, on page 296

This task generates an RSA key pair.

Note

- RSA keys are auto-generated at the time of router boot up. Hence, step 1 is required to be configured only if the RSA key-pair is missing on the router under some circumstances.
- The details of RSA and DSA keys are displayed in the running configuration.

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:

```
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.
- From Cisco IOS XR Release 7.3.2 onwards, you can configure this command from XR Config mode. For more details, see Public Key-Pair Generation in XR Config Mode, on page 144.

To delete the RSA keys, use the no form: **no crypto key generate rsa**

Step 2 **crypto key zeroize rsa** [**keypair-label**]

You can run the **crypto key zeroize** command only in the `exec` mode

Example:

```
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 you router. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys.
- To remove a specific RSA key pair, use the *keypair-label* argument.
- From Cisco IOS XR Release 7.3.2 onwards, you can delete key-pairs with the **no** form of the command in Step 1, on page 108 from XR Config mode. For more details, see Public Key-Pair Generation in XR Config Mode, on page 144.

Step 3 **show crypto key mypubkey rsa**

Example:

```
Router# show crypto key mypubkey rsa
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
```

(Optional) Displays the RSA public keys for your router.

The **show running-config** command also displays the RSA keys. The keys in the following example are in OpenSSL format.

Note
Only those keys that are generated in the `config` mode are visible in the running configuration.

```
Router(config)#crypto key generate rsa test
Router(config)#commit
Thu May 12 08:37:59.894 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nTbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOTycjsizI/TLBz9WoQX.rmxVwkNgTKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

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:

```
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:

```
Router# show crypto key mypubkey rsa
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
```

(Optional) Displays the RSA public keys for your router.

The **show running-config** command also displays the RSA keys. The keys in the following example are in OpenSSL format.

Note
Only those keys that are generated in the `config` mode are visible in the running configuration.

```
Router(config)#crypto key generate rsa test
Router(config)#commit
Thu May 12 08:37:59.894 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nTbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOTycjsizI/TLBz9WoQX.rmxVwkNgTKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
```

```
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

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.

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.

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:

```
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:

```
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:

```
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:

```
Router# show crypto ca certificates
```

(Optional) Displays information about the CA certificate.

CA enrollment URL

Table 11: Feature History Table

The CA Enrollment URL

- is a web address that routers use to connect with a Certification Authority (CA) for certificate enrollment and renewal, and
- automates the process of obtaining digital certificates, minimizing manual intervention and reducing the risk of potential errors.

How CA enrollment URL works

- When a router needs to enroll for a new certificate or renew an existing one, the router sends a request to the enrollment URL. The requests sent to the enrollment URL typically include the device's identity information and its public key.
- Upon receiving a request at the enrollment URL, the CA processes it to verify the identity of the requester. The verification involves checking the authenticity and validity of the information provided in the enrollment request.
- After the validation is successful, the CA issues a digital certificate. This certificate is then sent back to the router through the enrollment URL.
- The router then installs and use this certificate for secure communications.

IPv4 support for CA enrollment URL

The routers support IPv4 addresses or web addresses resolving to an IPv4 address as the CA server address when specifying the CA URL using the **enrollment url** command.

Guidelines for CA enrollment URL

These are the guidelines for CA enrollment URL:

- The enrollment URL string must start with http://CA_name, where CA_name is the host Domain Name System (DNS) name or IP address of the CA (for example, http://ca-server).

- If the CA CGI-bin script location is not /CGI-bin/pkiclient.exe at the CA (the default CA CGI-bin script location), you must also include the nonstandard script location in the URL in the form of http://CA-name/script-location, where script-location is the full path to the CA scripts.

Configure enrollment URL for CA

Use this procedure to enroll the URL for the CA.

Procedure

Step 1 Declare the CA and enter the trustpoint configuration mode using the **crypto ca trustpoint myca** command.

Example:

```
Router# configure
Router(config)# crypto ca trustpoint myca
```

Step 2 Specify the URL of the CA using the **enrollment url** command.

Example:

```
Router(config-trustp)# enrollment url http://ca.domain.com/certsrv/mscep/mscep.dll
```

Step 3 Execute the **show running-config** command to verify the enrollment URL for the CA.

Example:

```
Router# show running-config crypto ca trustpoint myca
crypto ca trustpoint myca
  enrollment url http://ca.domain.com/certsrv/mscep/mscep.dll
 !
```

Verify that the enrollment URL for the CA is http://ca.domain.com/certsrv/mscep/mscep.dll.

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:

```
Router#crypto ca trustpoint myca
```

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:

```
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:

```
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:

```
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:

```
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:

```
Router# show crypto ca certificates
```

Displays information about your certificate and the CA certificate.

The following example shows how to configure CA interoperability.

Comments are included within the configuration to explain various commands.

```
configure
hostname myrouter
domain name mydomain.com
end

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

crypto key generate rsa mykey

The name for the keys will be:mykey
Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose
Keypair
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus [1024]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]

show crypto key mypubkey rsa

Key label:mykey
Type     :RSA General purpose
Size     :1024
Created  :17:33:23 UTC Thu Sep 18 2003
Data     :
 30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00CB8D86
 BF6707AA FD7E4F08 A1F70080 B9E6016B 8128004C B477817B BCF35106 BC60B06E
 07A417FD 7979D262 B35465A6 1D3B70D1 36ACAFBD 7F91D5A0 CFB0EE91 B9D52C69
 7CAF89ED F66A6A58 89EEF776 A03916CB 3663FB17 B7DBEBF8 1C54AF7F 293F3004
 C15B08A8 C6965F1E 289DD724 BD40AF59 E90E44D5 7D590000 5C4BEA9D B5020301
 0001

! The following commands declare a CA and configure a trusted point.

configure
crypto ca trustpoint myca
enrollment url http://xyz-ultra5
enrollment retry count 25
enrollment retry period 2
rsakeypair mykey
end

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

! The following command authenticates the CA to your router.

crypto ca authenticate myca

Serial Number  :01
Subject Name   :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
Issued By      :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
```

```
Validity Start :07:00:00 UTC Tue Aug 19 2003
Validity End   :07:00:00 UTC Wed Aug 19 2020
Fingerprint:58 71 FB 94 55 65 D4 64 38 91 2B 00 61 E9 F8 05
Do you accept this certificate?? [yes/no]:yes

! The following command requests certificates for all of your RSA key pairs.

crypto ca enroll myca

% Start certificate enrollment ...
% Create a challenge password. You will need to verbally provide this
  password to the CA Administrator in order to revoke your certificate.
% For security reasons your password will not be saved in the configuration.
% Please make a note of it.

Password:
Re-enter Password:
    Fingerprint: 17D8B38D ED2BDF2E DF8ADBF7 A7DBE35A

! The following command displays information about your certificate and the CA certificate.

show crypto ca certificates

Trustpoint       :myca
============================================================
CA certificate
  Serial Number  :01
  Subject Name   :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Validity Start :07:00:00 UTC Tue Aug 19 2003
  Validity End   :07:00:00 UTC Wed Aug 19 2020
Router certificate
  Key usage      :General Purpose
  Status         :Available
  Serial Number  :6E
  Subject Name   :
        unstructuredName=myrouter.mydomain.com,o=Cisco Systems
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Validity Start :21:43:14 UTC Mon Sep 22 2003
  Validity End   :21:43:14 UTC Mon Sep 29 2003
  CRL Distribution Point
        ldap://coax-u10.cisco.com/CN=Root coax-u10 Certificate Manager,O=Cisco Systems
```

Certificate Authority Trust Pool Management

The trust pool feature is used to authenticate sessions, such as HTTPS, that occur between devices by using commonly recognized trusted agents called certificate authorities (CAs). This feature is enabled by default in the software to create a scheme to provision, store, and manage a pool of certificates from known CAs in a way similar to the services a browser provides for securing sessions. A special trusted point called a trust pool is designated, containing multiple known CA certificates from Cisco and possibly from other vendors. The trust pool consists of both built-in and downloaded CA certificates.

CA Certificate Bundling in the Trust Pool

The router uses a built-in CA certificate bundle that is packaged into the baseline image. 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

The use of the Certificate Authority requires that a crypto subsystem is included in the software image. Crypto is the Cisco proprietary encryption mechanism used in the Cisco software, which is available in the baseline image.

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 7.3.3, 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 121 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

Step 1 **crypto ca trustpool import url clean**

Example:

```
Router#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:

```
Router#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:

```
Router#show crypto ca trustpool

Trustpool: Built-In
================================================
CA certificate
  Serial Number  : 5F:F8:7B:28:2B:54:DC:8D:42:A3:15:B5:68:C9:AD:FF
  Subject:
        CN=Cisco Root CA 2048,O=Cisco Systems
Issued By      :
        CN=Cisco Root CA 2048,O=Cisco Systems
  Validity Start : 20:17:12 UTC Fri May 14 2004
  Validity End   : 20:25:42 UTC Mon May 14 2029
  SHA1 Fingerprint:
         DE990CED99E0431F60EDC3937E7CD5BF0ED9E5FA

Trustpool: Built-In
================================================
CA certificate
  Serial Number  : 2E:D2:0E:73:47:D3:33:83:4B:4F:DD:0D:D7:B6:96:7E
Subject:
        CN=Cisco Root CA M1,O=Cisco
  Issued By      :
```

```
      CN=Cisco Root CA M1,O=Cisco
Validity Start : 20:50:24 UTC Tue Nov 18 2008
Validity End   : 21:59:46 UTC Fri Nov 18 2033
SHA1 Fingerprint:
       45AD6BB499011BB4E84E84316A81C27D89EE5CE7
```

Displays the CA trust pool certificates of the router in a verbose format.

Retrieve CRL through the HTTP Proxy Server

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
Retrieve CRL through the HTTP Proxy Server	Release 7.3.1	CRL contains the serial numbers of the third-party certificates that are invalidated by the issuing Certificate Authority. In the event that the CRL Distribution point (CDP) is not directly reachable, you can fetch the CRL through the http proxy server using the newly introduced crypto ca http-proxy command. Command modified for this feature: crypto ca crl request

The router receives a certificate from a peer and downloads a CRL from the CA as part of certificate validation. The router then checks the CRL to make sure the certificate of the peer has not been revoked. If the certificate appears on the CRL, the router will not accept the certificate and will not authenticate the peer.

A CRL can be reused with the same certificate multiple times until the CRL expires.

If the router receives the certificate of a peer after the applicable CRL has expired, the router downloads the new CRL.

If the CRL Distribution point (CDP) is not directly reachable, you can obtain the CRL through the http proxy server using this feature.

Configuration Example

This example shows how to retrieve CRL through the http proxy server using the **crypto ca http-proxy** command for smart licensing:

```
<!----Enabling the Router to use HTTP Proxy Server to Retrieve CRL----!>

Router# config
Router(config)# crypto ca http-proxy 10.10.10.1 port 1
Router(config)# commit

<!----Registering the Router with a Token on the Smart Licensing Server----!>
```

```
Router# license smart register idtoken NWRkMTJjZjYtMzJhNi00YzYxLWI3M$
Router# commit
```

Verification

Smart licensing registration is validated by fetching the CRL from the CDP, through the http proxy server. If the validation is successful, then the **show crypto ca crls** command displays the CRLs. If the validation has failed, then the **show crypto ca crls** command displays no output.

This example shows how to verify the retrieved CRL and the license status:

```
<!----Verifying the Retrieved CRLs----!>

Router#show crypto ca crls
Thu Jun  6 13:43:00.763 UTC
CRL Entry
===============================================
  Issuer : CN=xyz-w2k Root CA 2,O=xyz Limited,C=BM
  Last Update : Dec 17 18:18:14 2018 GMT
  Next Update : Jun 15 18:18:14 2019 GMT
  CRL Distribution Point :
        http://xyz-w2k.cisco.com/CertEnroll/xyz-w2k-root.crl
CRL Entry
===============================================
  Issuer : CN=zxy-w2k SSL ICA G2,O=zxy,C=US
  Last Update : Jun  6 12:57:04 2019 GMT
  Next Update : Jun  9 12:57:04 2019 GMT
  CRL Distribution Point :
        http://zxy-w2k.cisco.com/CertEnroll/zxy-w2k-root.crl
RP/0/RP0/CPU0:ios#

<!----Verifying the License Status-----!>

Router#show license status
Smart Licensing is ENABLED
Utility:
  Status: DISABLED
Data Privacy:
  Sending Hostname: yes
    Callhome hostname privacy: DISABLED
    Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED
Transport:
  Type: Callhome
Registration:
  Status: REGISTERED
  Smart Account: BU Production Test 1
  Virtual Account:
  Export-Controlled Functionality: ALLOWED
  Initial Registration: SUCCEEDED on Jun 06 2019 13:42:46 UTC
  Last Renewal Attempt: None
  Next Renewal Attempt: Dec 03 2019 13:42:46 UTC
  Registration Expires: Jun 05 2020 13:37:45 UTC
License Authorization:
  Status: AUTHORIZED on Jun 06 2019 13:42:55 UTC
  Last Communication Attempt: SUCCEEDED on Jun 06 2019 13:42:55 UTC
  Next Communication Attempt: Jul 06 2019 13:42:54 UTC
  Communication Deadline: Sep 04 2019 13:37:55 UTC

Export Authorization Key:
  Features Authorized:
    <none>
```

Note If you want to fetch the latest CRL from a specific CDP, use the **crypto ca crl request** <*cdp-url*> [**http-proxy** <*ip-address*> **port** <*port-number*>] command.

Configuring Optional Trustpool Policy Parameters

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **crypto ca trustpool policy**

Example:

```
Router(config)#crypto ca trustpool policy
Router(config-trustpool)#
```

Enters ca-trustpool configuration mode where commands can be accessed to configure CA trustpool policy parameters.

Step 3 **cabundle url** **URL**

Example:

```
Router(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:

```
Router(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:

```
Router(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?

IOS -XR provides a mechanism by which a CA client sends a notification to a syslog server when certificates are on the verge of expiry. Alert notifications are sent either through the syslog server or Simple Network Management Protocol (SNMP) traps.

PKI traps retrieves the certificate information of the devices in the network. The device sends SNMP traps at regular intervals to the network management system (NMS) based on the threshold configured in the device.

An SNMP trap (certificate expiry notification) is sent to the SNMP server at regular intervals starting from 60 days to one week before the certificate end date. The notifications are sent at the following intervals:

The notifications are sent at the following intervals:

Intervals	Description	Notification Mode
First notification	The notification is sent 60 days before the expiry of the certificate.	The notificiation are in a warning mode.
Repeated notifications	The repeated notification is sent every week, until a week before the expiry of the certificate. The notifications are in a warning mode when the certificate is valid for more than a week.	The notifications are in a warning mode when the certificate is valid for more than a week.
Last notification	The notifications are sent every day until the certificate expiry date.	The notifications are in an alert mode when the validity of a certificate is less than a week.

The notifications include the following information:

- Certificate serial number
- Certificate issuer name
- Trustpoint name
- Certificate type
- Number of days remaining for the certificate to expire

- Certificate subject name

The following is a syslog message that is displayed on the device:

```
%SECURITY-CEPKI-1-CERT_EXPIRING_ALERT : Certificate expiring WITHIN A WEEK.
Trustpoint Name= check, Certificate Type= ID, Serial Number= 02:EC,
Issuer Name= CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN, Subject name= CN=cisco.com,
Time Left= 1 days, 23 hours, 59 minutes, 41 seconds
```

Restrictions for PKI Credentials Expiry Alerts

Alerts are not sent for the following certificates:

- Secure Unique Device Identifier (SUDI) certificates
- Certificates that belong to a trustpool. Trustpools have their own expiry alerts mechanism
- Trustpoint clones
- CA certificates that do not have a router certificate associated with it.
- Certificates with key usage keys

Restrictions for PKI Credentials Expiry Alerts

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, on page 126.

Regenerate the Certificate

The certificate becomes invalid once expired. When you see the certificate expiry notification, we recommend you to regenerate the certificate, as soon as possible.

Perform the following steps, to regenerate the certificates:

1. Clear the existing certificate using the following command:

   ```
   Router# clear crypto ca certificates [trustpoint-name]
   ```

   For example,

   ```
   Router# clear crypto ca certificates myca
   ```

2. We recommend you to regenerate a new keypair for the label configured under the trustpoint-name. The new keypair overwrites the old key pair.

   ```
   Router# crypto key generate rsa [keypair-lablel]
   ```

For example,

```
Router# crypto key generate rsa mykey
The name for the keys will be: mykey
% You already have keys defined for mykey
Do you really want to replace them? [yes/no]: yes
  Choose the size of the key modulus in the range of 512 to 4096 for your General Purpose
 Keypair. Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [2048]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]The name for the keys will be: mykey
% You already have keys defined for mykey
Do you really want to replace them? [yes/no]: yes
  Choose the size of the key modulus in the range of 512 to 4096 for your General Purpose
 Keypair. Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [2048]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

3. Reenroll the certificate using the following command. For more information, see Request Your Own Certificates, on page 114.

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

For example,

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

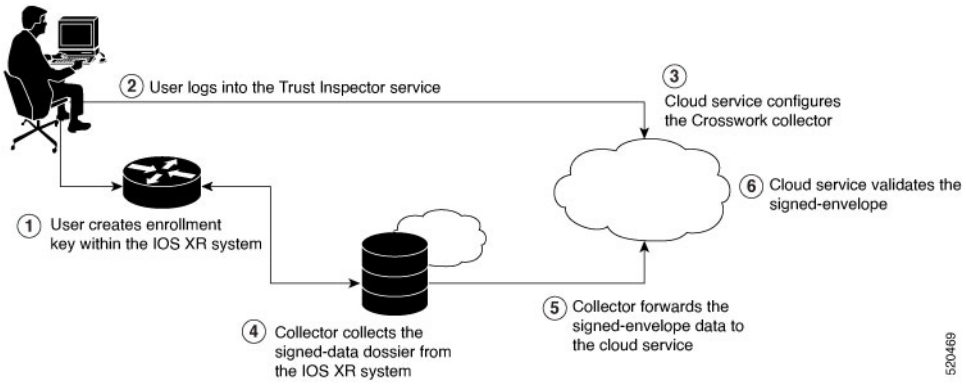
Integrating Cisco IOS XR and Crosswork Trust Insights

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 130
- Generate System Trust Point for the Leaf and Root Certificate, on page 132
- Generate Root and Leaf Certificates, on page 133
- System Certificates Expiry, on page 135
- Collect Data Dossier, on page 136

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)#commit
```

```
Router#configure
Router(config)#usergroup dossier-group
Router(config-ug)#taskgroup alltasks-dossier
Router(config-ug)#commit
```

```
Router#configure
Router(config)#username dossier-user
Router(config-un)#group dossier-group
Router(config-un)#commit
```

Configuration Example for for SSH and Netconf

```
Router#configure
Router(config)#ssh server v2
Router(config)#ssh server vrf default
Router(config)#ssh server netconf vrf default
Router(config)#netconf-yang agent
Router(config-ncy-agent)#ssh
Router(config-ncy-agent)#exit
Router(config)#domain name example.com
Router(config)#commit
```

Running Configuration

```
ssh server v2
ssh server vrf default
ssh server netconf vrf default
!
netconf-yang agent
 ssh
!
domain name example.com
```

While the dossier is collected from a device through SSH, the SSH session might timeout. Also, multiple ssh sessions to a device can result in the denial of some SSH sessions. To avoid such occurrence, the following configuration is recommended on the device:

```
Router#configure
Router(config)#ssh server rate-limit 600
Router(config)#line default
Router(config-line)#exec-timeout 0 0
Router(config-line)#session-timeout 0
Router(config-line)#commit
```

Running Configuration

```
ssh server rate-limit 600
!
line default
 exec-timeout 0 0
 session-timeout 0
!
```

Generate Key Pair

To enroll a system running Cisco IOS XR software, you must generate the key and the certificate for both the leaf and the root node. The system supports a two tier self-signed certificate chain for the enrollment key to support re-keying without re-enrollment of the certificate with the Crossworks service.

You can use the **system-root-key** and **system-enroll-key** options in the **crypto key generate** command to generate the root key and the enrollment key respectively, for all the hashing algorithms. You can do this for hashing algorithms such as RSA, DSA or ECDSA (including ECDSA nistp384 and ECDSA nitsp521).

To delete the RSA keys, use the no form: **no crypto key generate rsa**

The details of RSA and DSA keys are displayed in the running configuration.

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
```

You can also view the RSA keys in the running configuration. The keys in the following example are in OpenSSL format:

Note Only those keys that are generated in the `config` mode are visible in the running configuration.

```
Router(config)#crypto key generate rsa test
Router(config)#commit
Thu May 12 08:37:59.894 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nTbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOTycjsizI/TLBz9WoQX.rmxVwkNgTKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAgAAAAAAAA
AI81xZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

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. 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 125.

When you see the certificate expiry notification, we recommend you to regenerate the certificate, see Regenerate the Certificate, on page 126.

The following example shows how to regenerate the certificate.

```
Router# clear crypto ca certificates system-trustpoint
Router# crypto ca authenticate system-trustpoint
Router# crypto ca enroll system-trustpoint
```

Collect Data Dossier

Table 13: Feature History Table

Feature Name	Release Information	Description
Collect Filesystem Inventory	Release 7.3.1	With this feature, a snapshot of the filesystem metadata such as when the file was created, modified, or accessed is collected at each configured interval. In addition to displaying the changes that the file underwent as compared to the previous snapshot, the inventory helps in maintaining data integrity of all the files in the system.
IMA Optimization	Release 7.3.1	Integrity Measurement Architecture (IMA) is a Linux-based utility that attests and appraises the integrity of a system security, at runtime. In this release, IMA introduces the following IMA optimization aspects: • Incremental IMA that collects IMA events selectively and progressively instead of collecting all the IMA events at the same time. You can define the start of an IMA sequence, which consists of start event, start sequence number, and start time. • SUDI Signature - provides the hardware root of trust to the dossier that is collected by the system.

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 nonce 1234 include reboot-history
Thu Feb 27 22:20:57.542 IST
{"collection-start-time":1582822257.604932,"model-name":"http://cisco.com/ns/yang/Cisco-IOS-XR-ma","model-revision":"2019-08-05","license-udi":{"result-code":
 "Success", "license-udi": "UDI: PID:NCS-5501-SE,SN:FOC2107R0ZB\n"},"version":{"result-code":
 "Success", "version": "Cisco IOS XR Software, Version 7.0.12\nCopyright (c) 2013-2020 by
Cisco Systems, Inc.\n\nBuild Information:\n Built By     : user1\n Built On     : Mon Jan
27 01:36:26 PST 2020\n Built Host   : iox-lnx-076\n Workspace    : /auto/srca
rchive15/prod/7.0.12/cisco8000/ws\n Version      : 7.0.12\n Location     :
/opt/cisco/XR/packages/\n Label        : 7.0.12\n\ncisco NCS-5500 () processor\nSystem
uptime is 4 days 10 hours 12 minutes\n\n"},"platform":{"result-code": "Success", "platform":
 "Node              Type                        State             Config
state\n--------------------------------------------------------------------------------\n0/RP0/CPU0
        NCS-5501-SE(Active)        IOS XR RUN        NSHUT\n0/RP0/NPU0        Slice

            UP               \n0/FT0            NCS-1RU-FAN-FW             OPERATIONAL
      NSHUT\n0/FT1            NCS-1RU-FAN-FW             OPERATIONAL       NSHUT\n0/PM0
           NCS-1100W-ACFW             FAILED            NSHUT\n0/PM1
NCS-1100W-ACFW             OPERATIONAL
NSHUT\n"},"reboot-history":{"result-code":"Success","model-name":"Cisco-IOS-XR-linux-os-reboot-history-oper","model-revision":"2019-04-05","node":[{"node-name":
 "0/RP0/CPU0", "reboot-history": [{"reason": "
User initiated graceful reload", "time": "Wed Feb 19 15:25:11 2020", "cause-code": 1, "no":
 1}, {"reason": "CARD_SHUTDOWN", "time": "Wed Feb 19 16:38:00 2020", "cause-code": 37, "no":
 2}, {"reason": "CARD_SHUTDOWN", "time": "Wed Feb 19 19:06:27 2020", "cause-code": 37, "no":
 3}, {"reason": "CARD_SHUTDOWN", "time": "Thu Feb 20 11:50:50 2020", "cause-code": 37, "no":
 4}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21 10:54:09 2020", "cause-code": 37, "no":
 5}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21
19:00:10 2020", "cause-code": 37, "no": 6}, {"reason": "CARD_SHUTDOWN", "time": "Sun Feb
23 12:05:25 2020", "cause-code": 37, "no": 7}]}, {"node-name": "0/0/CPU0", "reboot-history":
 [{"reason": "Reboot triggered by install", "time": "Tue Feb  4 19:59:23 2020", "cause-code":
 36, "no": 1}, {"reason": "CARD_SHUTDOWN", "time": "Tue Feb  4 20:12:06 2020", "cause-code":
 37, "no": 2}, {"reason": "Headless SDR", "time": "Sun Feb  9 17:45:25 2020", "cause-code":
 671088647, "no": 3}, {"reason": "User initiated graceful reload", "time": "Sun Feb  9
17:45:29 2020", "cause-code": 241, "no": 4}, {"reason": "CARD_SHUTDOWN", "time": "Sun Feb
 9 18:28:25 2020", "cause-code": 37, "no": 5}, {"reason": "Headless SDR", "time": "Sun Feb
  9 19:01:55 2020", "cause-code": 671088647, "no": 6}, {"reason": "Headless SDR", "time":
"Wed Feb 19 15:25:19 2020", "cause-code": 671088647, "no": 7}, {"reason": "CARD_SHUTDOWN",
 "time": "Wed Feb 19 16:37:46 2020", "cause-code": 37, "no": 8}, {"reason": "CARD_SHUTDOWN",
 "time": "Wed Feb 19 19:06:1
4 2020", "cause-code": 37, "no": 9}, {"reason": "CARD_SHUTDOWN", "time": "Thu Feb 20 11:50:37
 2020", "cause-code": 37, "no": 10}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21 10:54:01
 2020", "cause-code": 37, "no": 11}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21 18:59:57
 2020", "cause-code": 37, "no": 12}, {"reason": "CARD_SHUTDOWN", "time": "Sun Feb 23 12:05:12
 2020", "cause-code": 37, "no": 13}]}]},"collection-end-time":1582822260.296664}
Router#
```

Collect Filesystem Inventory

The metadata of the filesystem can be collected using data dossier. The metadata of the file includes information about time the file was created, last accessed, last modified and so on. A snapshot is captured at each configured interval. The initial snapshot shows a complete snapshot of all files in the filesystem. The files are scanned periodically and new inventory data is collected and stored as incremental snapshots.

To enable this feature, use the **filesystem-inventory** command.

```
Router(config)#filesystem-inventory
Router(config-filesystem-inventory)#snapshot-interval 2
Router(config-filesystem-inventory)#commit
```

The `snapshot-interval` is the time interval in 15-minute blocks. The interval ranges 1–96. For example, value of `2` indicates that a snapshot interval is collected every 30 minutes. The snapshots are stored in./`misc/scratch/filesysinv` The logs are stored in `/var/log/iosxr/filesysinv/*`.

To retrieve the filesystem inventory, use the following dossier command. Output is presented in JSON format.

```
show platform security integrity dossier include filesystem-inventory | file
<platform>-parent.json

{"collection-start-time":1610168028.380901,
"model-name":"http://cisco.com/ns/yang/Cisco-IOS-XR-ama",
"model-revision":"2019-08-05","license-udi":{"result-code": "Success", "license-udi":
"UDI: PID:NCS-55A1-24H,SN:FOC2104R15R\n"},"version":{"result-code": "Success",
"version": "Cisco IOS XR Software, Version 7.3.1
\nCopyright (c) 2013-2020 by Cisco Systems, Inc.\n\nBuild Information:\n
Built By     : <user>\n Built On     : Thu Jan  7 17:16:02 PST 2021\n
Built Host   : <host>\n Workspace    : <ws>
Version     : 7.3.1\n  Location    : /opt/cisco/XR/packages/\n Label       : 7.3.1\n\ncisco

() processor\nSystem uptime is 8 hours 7 minutes\n\n"},"platform":{"result-code":
"Success", "platform":
"Node              Type                        State             Config state
-------------------------------------------------------------------------------
0/RP0/CPU0        <node-type>(Active)       IOS XR RUN        NSHUT\n
0/RP0/NPU0        Slice                      UP
0/RP0/NPU1        Slice                      UP
0/FT0             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
0/FT1             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
0/FT2             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
PM1               <platform>-1100W-ACRV      OPERATIONAL       NSHUT
"},
----------------------------Output is snipped for brevity
-------------------------------------
```

To limit the number of snapshots, use the following command:

```
show platform security integrity dossier include filesystem-inventory
filesystem-inventory-options '{\"0/RP0/CPU0\": {\"block_start\": 0, \"count\": 1}}'
```

To start from a new block, use the following command:

```
show platform security integrity dossier include filesystem-inventory
filesystem-inventory-options '{\"0/RP0/CPU0\": {\"block_start\": 5}}'
```

To collect data from a remote node, use the following command:

```
show platform security integrity dossier include filesystem-inventory
filesystem-inventory-options '{\"0/RP1/CPU0\": {\"block_start\": 0}}' | file
harddisk:PE1_remote.json
```

Incremental Integrity Measurement Architecture

With incremental Integrity Meansurement Architecture (IMA), you can define the starting IMA sequence that you want to include in a response. The system then starts to report the subsequent events.

```
show platform security integrity dossier incremental-ima
"{\"ima_start\":[{\"0/RP0/CPU0\":{\"start_event\":1000,\"start_time\":\"Tue Feb 16 09:15:17
 2021\"}}]}"
```

Associated Command

- **show platform security integrity dossier**
- **utility sign**

Procedure to Test Key Generation and Data-signing with Different Key Algorithm

You can follow these steps to test key generation and data-signing with a different key algorithm:

- Unconfigure the trustpoint (using the **no crypto ca trustpoint system-trustpoint** command)
- Clear the certificates that were generated earlier (using the **clear crypto ca certificates system-trustpoint** command)
- Generate new keys.
- Configure the system trustpoint again.
- Authenticate and enroll the system trustpoint to generate the certificates.

See How to Integrate Cisco IOS XR and Crosswork Trust Insights, on page 128 section for configuration steps of each task.

Verify Authenticity of RPM Packages Using Fingerprint

Table 14: Feature History Table

Feature Name	Release Information	Description
Verify Authenticity of RPM Packages Using Fingerprint	Release 7.3.1	This feature helps in verifying the authenticity of an installable package using fingerprint values. The fingerprint value of the package is compared with a point of reference called Known Good Value (KGV). The KGV for an image or package is generated after it is built by Cisco. After installing the package, the associated install time and build time fingerprint values are compared using Yang RPC to determine whether the package is genuine. A match in the fingerprints indicates that the package published on CCO and that installed on router are the same.

Is there a simple way to determine the authenticity of a package that is installed on a router? Is there a mechanism to identify whether a package signature is checked at install time, or detect changes to the files after the package is installed at run time?

Cisco IOS XR, Release 7.3.1 introduces a fingerprint mechanism to verify the authenticity of a package that Cisco releases. This mechanism helps determine whether the installed package is genuine, where the installed and running software matches the software that is published by Cisco.

There are significant security measures for installing software using GPG and IMA signing. However, there is need to report more data for Cisco Crosswork application to monitor and flag potential issues for further investigation. Cisco Crosswork monitors the installed software over a period to help accomplish the following tasks:

- To determine whether there are any differences between the software that is published on Cisco.com and that downloaded to the router.
- To determine whether any files in a package have been altered, either accidentally or maliciously, from the time the package was installed.

A Known Good Value (KGV) is calculated and published for each package. This value is considered the right value for the package.

Two fingerprint (hex) values for each active or committed packages are monitored to ensure authenticity of the package:

- **Install time fingerprint:** Hex value that represents the software in the package at install time. An RPM is genuine if it is not modified before install, and it matches the KGV. Whereas a manipulated RPM shows a mismatch in the fingerprint that is published in the KGV.
- **Run time fingerprint:** Hex value that represents the running software of an installed package. The value matches the corresponding install time fingerprint if the RPM has not been modified since the install time. If there are changes to the files, the run time and install time fingerprints show a mismatch. Every time the files that are installed by an RPM are changed, the run time fingerprint also changes. A value of `0` (zero) is displayed if no run time fingerprint is available for a package. This is used to monitor changes to the running software over time.

Note These two values are displayed only in the Yang model output. No CLI commands are provided to view these values.

```
Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:97f5bc36-0eb0-4d2f-9c6f-3d34fea14be0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
    <install xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spirit-install-instmgr-oper">
      <packages>
        <active>
         <summary>
          <rpm-fingerprint-status>generation-up-to-date</rpm-fingerprint-status>
            <rpm-fingerprint-timestamp>Mon Jun 15 15:58:22 2020</rpm-fingerprint-timestamp>

             <package>
              <name>asr9k-xr</name>
               <version>7.3.1</version>
               <release>r731</release>
```

```
              <gpg-key-id>ddcead3dcb38048d</gpg-key-id>
               <rpm-fingerprint>

<rpm-fingerprint-install-time>2871bf68d3cd764938775afc9e5a69c130f9fbde</rpm-fingerprint-install-time>


<rpm-fingerprint-run-time>2871bf68d3cd764938775afc9e5a69c130f9fbde</rpm-fingerprint-run-time>

              </rpm-fingerprint>
             </package>

            <package>
             <name>asr9k-mcast-x64</name>
             <version>2.0.0.0</version>
             <release>r731</release>
             <gpg-key-id>ddcead3dcb38048d</gpg-key-id>
              <rpm-fingerprint>

<rpm-fingerprint-install-time>3ddca55bc00a0ce2c2e52277919d398621616b28</rpm-fingerprint-install-time>


<rpm-fingerprint-run-time>3ddca55bc00a0ce2c2e52277919d398621616b28</rpm-fingerprint-run-time>

              </rpm-fingerprint>
             </package>
------------------------------ Truncated for brevity ---------------------------------
```

In the example, both the install time and run time fingerprints are the same.

The fingerprint generation status is used to indicate how up-to-date the run time fingerprints are. This may indicate that generation is currently in progress and will complete shortly, or generation is awaiting the end of an atomic change.

Support for Ed25519 Public-Key Signature System

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Support for Ed25519 Public-Key Signature System	Release 7.3.1	This feature allows you to generate and securely store crypto key pair for the Ed25519 public-key signature algorithm on Cisco IOS XR 64-bit platforms. This signature system provides fast signing, fast key generation, fool proof session keys, collision resilience, and small signatures. The feature also facilitates integration of Cisco IOS XR with Cisco Crosswork Trust Insights. Commands introduced for this feature are: • crypto key generate ed25519 • crypto key zeroize ed25519 • show crypto key mypubkey ed25519 Commands modified for this feature are: • ca-keypair • keypair

The Cisco IOS XR Software Release 7.3.1 introduces the support for Ed25519 public-key signature algorithm on 64-bit platforms. Prior to this release, only DSA, ECDSA, and RSA signature algorithms were supported. The Ed25519 signature algorithm uses the elliptic curve cryptography that offers a better security with faster performance when compared to other signature algorithms.

You can generate the Ed25519 crypto keys either with an empty label or with two predefined labels: **system-root-key** and **system-enroll-key**. In the case of an empty label, the system generates the key pair against the default label. You can use the key pairs with the predefined labels to integrate Cisco IOS XR with Cisco Crosswork Trust Insights.

Generate Crypto Key for Ed25519 Signature Algorithm

Configuration Example

To generate the Ed25519 crypto key, use the **crypto key generate ed25519** command in XR EXEC mode or XR Config 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.

Note From Cisco IOS XR Release 7.3.2 onwards, you can generate and delete key-pairs from XR Config mode, as well. For more details, see Public Key-Pair Generation in XR Config Mode, on page 144.

Verification

Use the **show crypto key mypubkey ed25519** command to view all Ed25519 crypto keys generated on the system.

```
Router# show crypto key mypubkey  ed25519

Mon Nov 30 07:05:06.532 UTC
Key label: the_default
Type : ED25519
Size : 256
Created : 07:03:17 UTC Mon Nov 30 2020
Data :
FF0ED4E7 71531B3D 9ED72C48 3F79EC59 9EFECCC3 46A129B2 FAAA12DD EE9D0351

]
```

Related Topics

- Support for Ed25519 Public-Key Signature System, on page 142
- Integrate Cisco IOS XR with Cisco Crosswork Trust Insights using Ed25519, on page 143

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 127.

Public Key-Pair Generation in XR Config Mode

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Public Key-Pair Generation in XR Config Mode	Release 7.3.2	This feature allows you to generate public-key pairs in the XR Config mode, which in turn lets you save configurations. You can then load these saved configurations across different routers to quickly deploy the key-pair configurations. You could generate public-key pairs in earlier releases only in the XR EXEC mode, which does not save configurations. So manually executing the key-pair generation commands on every router was time-consuming. The following commands are available in XR Config mode, in addition to XR EXEC mode: • crypto key generate rsa • crypto key generate dsa • crypto key generate ecdsa • crypto key generate ed25519

Public Key-Pair Generation in XR Config mode supports the following key-types and key sizes in FIPS (Federal Information Processing Standard) and non-FIPS modes.

Table 17: Supported Key-Types for non-FIPS and FIPS mode

Keys-Types	Non-FIPS mode	FIPS mode
RSA	Supported for all key sizes from 512 - 4096	Supported for key sizes 2048, 3072, 4096
DSA	Supported for key sizes 512, 768, 1024	Supported for key size 2048
ECDSA	Supported for key sizes nistp256, nistp384,nistp512	Supported for key sizes nistp256, nistp384,nistp512
ED25519	Supported	Not Supported

For more details on FIPS, see Configuring FIPS Mode, on page 293 chapter in this guide.

Guidelines and Restrictions:

The following guidelines and restrictions apply for generating crypto keys-pairs in XR Config mode:

- This feature doesn’t support generation of generation of **system-root-key** and **system-enroll-key**.
- The key-pairs generated in XR Config mode overwrites any previously generated key-pairs in XR EXEC mode.
- The router doesn't support overwriting key-pairs generated in XR Config mode from XR EXEC mode.
- When you execute **no** form of the **crypto key generate** commands in XR Config Mode, it deletes only those keys generated in XR Config mode.
- The router doesn't support deleting key-pairs generated in XR Config mode from XR EXEC mode.
- When you execute the **crypto key generate** commands in XR EXEC mode, it doesn't overwrite or delete keys generated in XR Config mode.
- The show command **show crypto key mypubkey** displays the keys generated in XR EXEC mode first, followed by the keys generated in XR Config mode.

Configuration Examples:

The following examples show the creation of key-pairs in XR Config mode:

```
Router# conf t
Router(config)#crypto key generate dsa 512
Router(config)#crypto key generate rsa user1 general-keys 2048
Router(config)#crypto key generate rsa user2 usage-keys 2048
Router(config)#crypto key generate rsa 2048
Router(config)#crypto key generate ecdsa nistp256
Router(config)#crypto key generate ecdsa nistp384
Router(config)#crypto key generate ecdsa nistp521
Router(config)#crypto key generate ed25519
Router(config)#commit
```

Use **no** form of the command in XR Config mode to delete any of the key-pairs.

System Logs and Error Messages:

The router generates these system logs on successful creation of key-pairs:

```
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key DSA generated, label:the_default,
modBits:1024
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key ECDSA_NISTP256 generated,
label:the_default, modBits:256
```

The router generates these system logs on deletion of key-pairs:

```
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key RSA zeroized, label:user1
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key DSA zeroized, label:the_default
```

The router generates these error messages if you try to overwrite the key-pairs generated in XR Config Mode from XR EXEC mode:

```
Router#conf t
Router(config)#crypto key generate ed25519
Router(config)#commit
Router#crypto key generate ed25519
Cannot execute the command : Operation not permitted
ce_cmd[68727]: %SECURITY-CEPKI-6-ERR_2 : Cannot      execute the command :   Operation not
 permitted
ce_cmd[68736]: %SECURITY-CEPKI-6-ERR : Key is added as part of config mode, key deletion
is not allowed , delete key from config mode
```

The router generates these error messages if you try to delete key-pairs generated in XR Config Mode from XR EXEC mode:

```
Router#conf t
Router(config)#crypto key generate ed25519
Router(config)#commit
Router#crypto key zeroize ed25519
Cannot execute the command : Operation not permitted
ce_cmd[68736]: %SECURITY-CEPKI-6-ERR_2 : Cannot execute the command :   Operation not
permitted
```

To View the Generated Key-Pairs:

You can view the key-pairs generated in XR Config mode, listed under **Public keys from config sysdb** in the following command output:

```
Router#show crypto key mypubkey ecdsa
Key label: the_default
Type     : ECDSA General Curve Nistp256
Degree   : 256
Created  : 11:49:22 IST Wed Apr 21 2021
Data     :
 04D6D132 2253ABD0 81449E3F 9D5CEA3A 1107950A 829E9090 8960FBD5 ABA039B7
 24A4E217 7EA47475 91C60AC7 013DBC2E EA8434D9 0BD5B0FC 694913AE 0098A4F5
 77

Key label: the_default
Type     : ECDSA General Curve Nistp521
Degree   : 521
Created  : 22:44:22 IST Thu Mar 18 2021
Data     :
 04017798 4369F493 8D0E57D1 1975FC46 CDC03A78 03A9F90E B38CA504 17DB9A64
 D1DEA6A6 D23E7E20 4D8D4D31 C7878BDB BF5EEE40 1978A889 70C5D703 BB033B77
 0FFD9201 366A9AC8 35E69BB3 97FF4E91 6B498510 39425971 C5E43858 83286088
 A6A7BF92 0EA2B416 BD4E81CE DCEB65F1 15CC75B5 91204E89 3339A168 2382CAB6
```

```
 40170131 8F


------------------------------
 Public keys from config sysdb:
 ------------------------------
 Key label: the_default
Type     : ECDSA General Curve Nistp384
Degree   : 384
Created  : 11:51:52 IST Wed Apr 21 2021
Data     :
 045F7C14 1A88C27E 9CED3FF1 7FEDFA03 B49575FA 7AD88370 BC9C7D7F F99C8917
 33620916 758BDEFC 7187E33A 2D3CCD33 14FF3267 9855A5E9 E3BD166C CE838462
 40742231 6198EE12 3E189F42 22A8149A 8E7B186D 88E728D4 7F47D565 53441061
 79
```

Information About Implementing Certification Authority

Supported Standards for Certification Authority Interoperability

Cisco supports the following standards:

- 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.

Note The Internet Key Exchange (IKE) standard is not supported.

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.

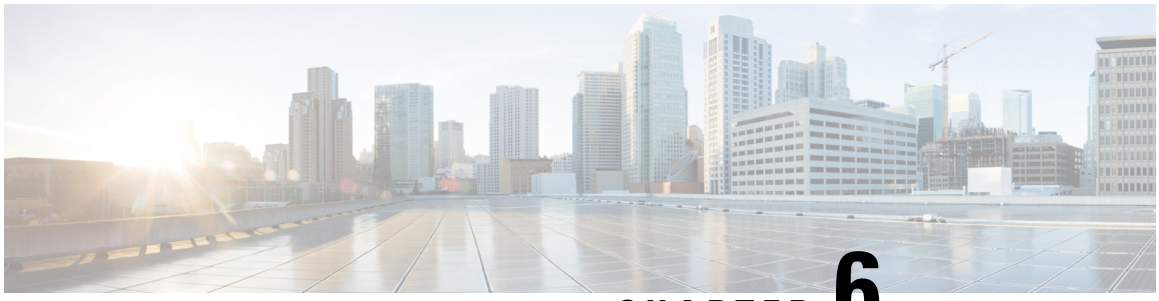
Digital certificates provide the link. A digital certificate contains information to identify a user or device, such as the name, serial number, company, department, or IP address. It also contains a copy of the entity's public key. The certificate is itself signed by a CA, a third party that is explicitly 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 6

Implementing Keychain Management

This module describes how to implement keychain management on Cisco 8000 Series Routers. 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.

- Prerequisites for Configuring Keychain Management, on page 149
- Restrictions for Implementing Keychain Management, on page 149
- Information About Implementing Keychain Management, on page 149
- How to Implement Keychain Management, on page 150
- Configuration Examples for Implementing Keychain Management, on page 158

Prerequisites for Configuring Keychain Management

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.

Restrictions for Implementing Keychain Management

You must be aware that changing the system clock impacts the validity of the keys in the existing configuration.

Information About Implementing Keychain Management

The keychain by itself has no relevance; therefore, it must be used by an application that needs to communicate by using the keys (for authentication) with its peers. The keychain provides a secure mechanism to handle the keys and rollover based on the lifetime. Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), and Intermediate System-to-Intermediate System (IS-IS) use the keychain to implement a hitless key rollover for authentication. BGP uses TCP authentication, which enables the authentication option and sends the Message Authentication Code (MAC) based on the cryptographic algorithm configured for the keychain. For information about BGP, OSPF, and IS-IS keychain configurations, see the *Configure>Routing* end-user guide listed here.

- Resource Reservation Protocol (RSVP) uses keychain for authentication.

For more information about RSVP, see the *Configure>MPLS* end-user guide listed here.

- IP Service Level Agreements (IP SLAs) use a keychain for MD5 authentication for the IP SLA control message. For more information about IP SLAs, see the *Configure>System Monitoring* end-user guide listed here.

To implement keychain management, you must understand the concept of key lifetime, which is explained in the next section.

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.

How to Implement Keychain Management

This section contains the following procedures:

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:

```
Router(config)# key chain isis-keys
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 XR Config mode attributes or keychain-key configuration mode attributes (for example, lifetime or key string).

Step 3 **commit**

Commits the configuration changes and remains within the configuration session.

Step 4 **show key chain** *key-chain-name*

Example:

```
Router# show key chain isis-keys
Key-chain: isis-keys/ -

accept-tolerance -- infinite
Key 8 -- text "1104000E120B520005282820"
  cryptographic-algorithm -- MD5
  Send lifetime:   01:00:00, 29 Jun 2006 - Always valid  [Valid now]
  Accept lifetime: 01:00:00, 29 Jun 2006 - Always valid [Valid now]
```

(Optional) Displays the name of the keychain.

Note
The *key-chain-name* argument is optional. If you do not specify a name for the *key-chain-name* argument, all the keychains are displayed.

Step 5 **show run**

Example:

```
key chain isis-keys
 accept-tolerance infinite
 key 8
  key-string mykey91abcd
  cryptographic-algorithm MD5
  send-lifetime 1:00:00 june 29 2006 infinite
```

```
  accept-lifetime 1:00:00 june 29 2006 infinite
  !
 !
!
```

Configure Tolerance Specification to Accept Keys

This task configures the tolerance specification to accept keys for a keychain to facilitate a hitless key rollover for applications, such as routing and management protocols.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

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

Example:

```
Router(config-isis-keys)# accept-tolerance infinite
```

Configures an accept tolerance limit—duration for which an expired or soon-to-be activated keys can be used for validating received packets—for a key that is used by a peer.

- Use the *value* argument to set the tolerance range in seconds. The range is from 1 to 8640000.
- Use the **infinite** keyword to specify that an accept key is always acceptable and validated when used by a peer.

Step 4 **commit**

Commits the configuration changes and remains within the configuration session.

Configure Key Identifier for Keychain

This task configures a key identifier for the keychain.

You can create or modify the key for the keychain.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
Router(config-isis-keys)# key 8
```

Creates a key for the keychain. The key ID has to be unique within the specific keychain.

- Use the *key-id* argument as a 48-bit integer.

Step 4 **commit**

Commits the configuration changes and remains within the configuration session.

Configure Text for Key String

This task configures the text for the key string.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **key chain** *key-chain-name*

Example:

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
Router(config-isis-keys)# key 8
Router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **key-string** [**clear** | **password**] *key-string-text*

Example:

```
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 **commit**

Commits the configuration changes and remains within the configuration session.

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:

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
Router(config-isis-keys)# key 8
Router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **accept-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]

Example:

```
Router(config-isis-keys)# key 8
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 **commit**

Commits configuration changes and exits the configuration session.

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:

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
Router(config-isis-keys)# key 8
Router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

Step 4 **send-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]

Example:

```
Router(config-isis-keys)#key 8
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 **commit**

Commits the configuration changes and remains within the configuration session.

Configure Cryptographic Algorithm

This task allows the keychain configuration to accept the choice of the cryptographic algorithm.

From Cisco IOS XR Software 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:

```
Router(config)# key chain isis-keys
Router(config-isis-keys)#
```

Creates a name for the keychain.

Step 3 **key** *key-id*

Example:

```
Router(config-isis-keys)# key 8
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:

```
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 **commit**

Commits configuration changes and exits the configuration session

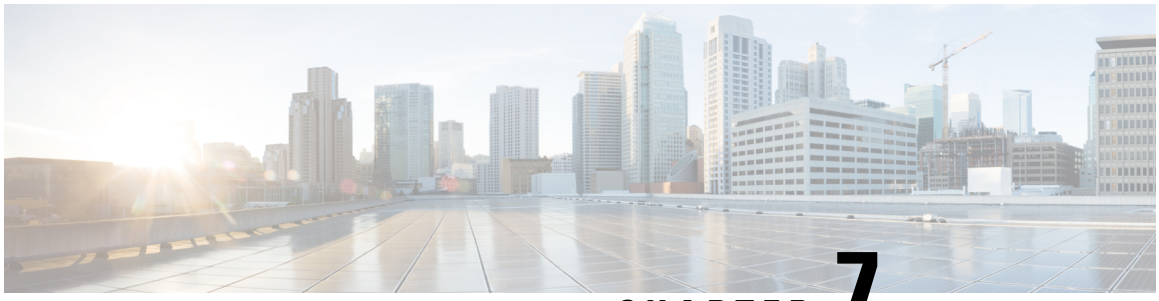
Configuration Examples for Implementing Keychain Management

This section provides the following configuration example:

Configuring Keychain Management: Example

The following example shows how to configure keychain management:

```
configure
 key chain isis-keys
  accept-tolerance infinite
  key 8
   key-string mykey91abcd
   cryptographic-algorithm MD5
   send-lifetime 1:00:00 june 29 2006 infinite
   accept-lifetime 1:00:00 june 29 2006 infinite


Router#show key chain isis-keys

Key-chain: isis-keys/ -

accept-tolerance -- infinite
Key 8 -- text "1104000E120B520005282820"
  cryptographic-algorithm -- MD5
  Send lifetime:   01:00:00, 29 Jun 2006 - Always valid  [Valid now]
  Accept lifetime: 01:00:00, 29 Jun 2006 - Always valid [Valid now]
```



CHAPTER 7

Configuring MACsec

This module describes how to configure Media Access Control Security (MACsec) encryption on Cisco 8000 Series Routers. MACsec is a Layer 2 IEEE 802.1AE standard for encrypting packets between two MACsec-capable routers.

- Understanding MACsec Encryption, on page 159
- MKA Authentication Process, on page 160
- MACsec Frame Format, on page 161
- Advantages of Using MACsec Encryption, on page 161
- Hardware Support for MACsec, on page 162
- MACsec PSK, on page 162
- Fallback PSK, on page 163
- Configuring and Verifying MACsec Encryption , on page 164
- Create a MACsec keychain, on page 166
- Create a user-defined MACsec policy, on page 170
- EAPoL Ether-type and destination address, on page 172
- Applying MACsec Configuration on an Interface, on page 173
- MACsec mode on PHY, on page 182
- MACsec Policy Exceptions, on page 183
- Verifying MACsec Encryption on IOS XR, on page 185
- Verifying MACsec Encryption on Cisco 8000 Series Routers , on page 195
- MACsec SecY Statistics, on page 199
- Power-on Self-Test KAT for Common Criteria and FIPS, on page 207
- Secure Key Integration Protocol, on page 210
- Related Commands for MACsec, on page 217

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.

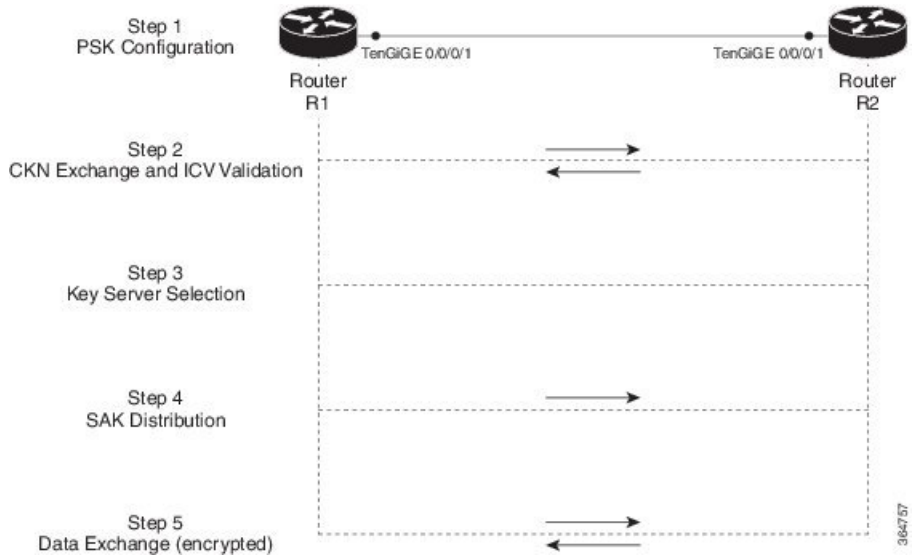
MACsec secures data on physical media, making it impossible for data to be compromised at higher layers. As a result, MACsec encryption takes priority over any other encryption method such as IPsec and SSL at higher layers. MACsec is configured on the Customer Edge (CE) router interfaces that connect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

MKA Authentication Process

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 7: 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).

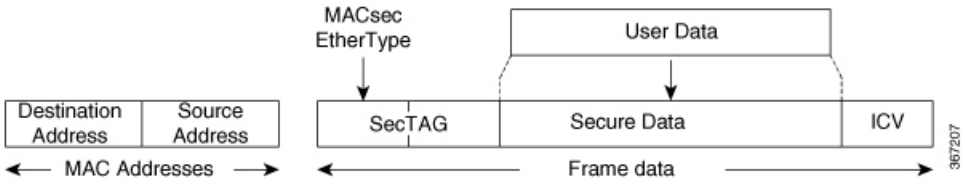
Step 4: A security association is formed between the peers. The key server generates and distributes the Secure Association Key (SAK) to the key client (peer). Each secure channel is supported by an overlapped sequence of Security Associations (SA). Each SA uses a new Secure Association Key (SAK).

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 8: 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

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
MACsec capability for 88-LC0-36FH-M	Release 7.3.15	The Cisco 8800 36x400GE QSFP56-DD Line Card with MACsec based on Q200 Silicon (88-LC0-36FH-M) supports point-to-point (P2P) MACsec capability with 400GE line rate encryption on its physical interfaces. This release also introduces point-to-multipoint (P2MP) MACsec capability on both 8800-LC-36FH-M and 8800-LC-48H (Cisco 8800 48x100 GbE QSFP28 Line Card based on Q100 Silicon). The P2MP topology connects multiple nodes to one central node, thereby avoiding unnecessary packet replication at the ingress router.

The MACsec technology is supported on 48-port 100GE line cards and 36-port 400GE line cards.

Table 19: MACsec Hardware Support Matrix

Cisco IOS XR Software Release	Product ID (PID)
Release 7.5.2	8202-32FH-M
Release 7.3.3	88-LC0-34H14FH
Release 7.3.15	88-LC0-36FH-M
Release 7.0.12	8800-LC-48H

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 (P2P) 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 Applying MACsec Configuration on an Interface, on page 173.

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.
- In P2P topologies, a rollover to the fallback PSK happens when either of the nodes in the Secure Association (SA) cannot peer up with the primary PSK. Whereas, in P2MP, the fallback happens only at the expiry or deletion of the primary key on all peers, not just on one of the peers. On deletion or expiry of the primary PSK on one of the nodes, say R1, a new key server is chosen among the peer nodes that does a SAK rekey for the remaining nodes. This ensures that R1 is no longer part of the SA, and the network drops all traffic to and from R1.

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 Applying MACsec Configuration on an Interface, on page 173.

Active Fallback

The Cisco IOS XR Software Release 7.0.14 introduces the support for active fallback feature that initiates a fallback MKA session on having fallback configuration under the interface.

The key benefits of active fallback feature are:

- Faster session convergence on fallback, in the event of primary key deletion, expiry or mismatch.
- Faster traffic recovery under should-secure security policy when both primary and fallback mismatch happens.

With the introduction of active fallback functionality, the output of various MACsec show commands include the fallback PSK entry as well. If the session is secured with primary key, the fallback session will be in ACTIVE state. See, Verifying MACsec Encryption on IOS XR, on page 185 for details and sample outputs.

Note If the peer device is running on an older release that does not support active fallback feature, you must configure the **enable-legacy-fallback** command under the macsec-policy to ensure backward compatibility.

Configuring and Verifying MACsec Encryption

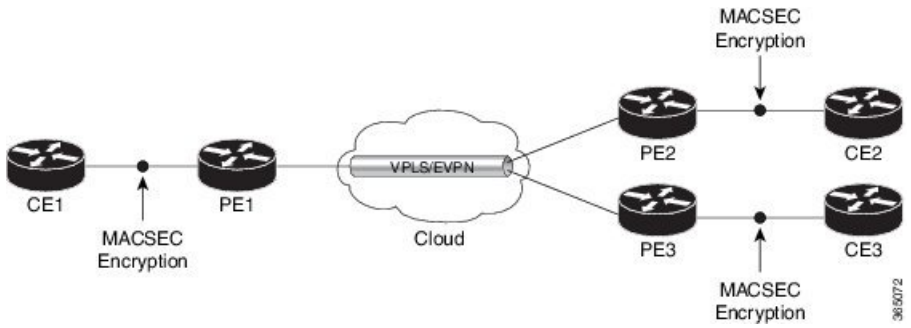
MACsec can be configured on physical Ethernet interfaces or interface bundles (link bundles), as explained in this section.

Note Enabling MACsec encryption on any on physical Ethernet interfaces or interface bundles (link bundles) will add an overhead of 32 bytes on the maximum transmission unit (MTU) size of link-state packets (LSPs). Therefore, you must set the LSP MTU to 32 bytes less than the interface MTU, to account for MACsec overhead. Use the **lsp-mtu** command to configure the maximum transmission unit (MTU) size of link-state packets (LSPs) on each router where MACsec is enabled.

Use Case 1: MACsec in a VPLS/EVPN

A typical VPLS network often suffers the injection of labeled traffic from potential hackers. The following figure illustrates the use of MACsec in a VPLS/EVPN network for encrypting the data being exchanged over the VPLS cloud. In this topology MACsec is configured on the PE-facing interfaces of the CE routers.

Figure 9: MACSec in a VPLS/EVPN Cloud



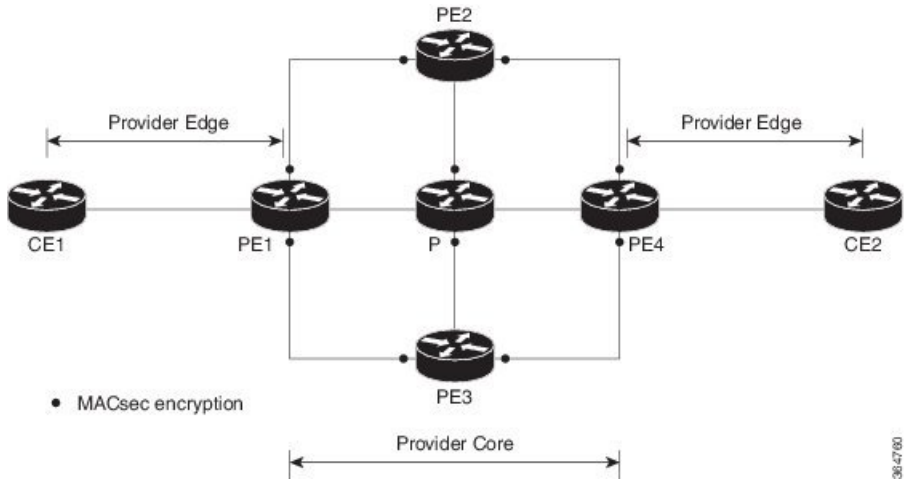
Use Case 2: MACsec in an MPLS Core Network

MACsec in an MPLS core network can be configured on physical interfaces or link bundles (Link Aggregation Group or LAG).

In the following topology, MACsec is configured on all router links in the MPLS core. This deployment is useful when the MPLS network spans data centers that are not co-located in the same geography. Each link is, therefore, a link between two data centers and all data exchanged is encrypted using MACsec.

The following figure illustrates the use of MACsec on physical interfaces in an MPLS core network.

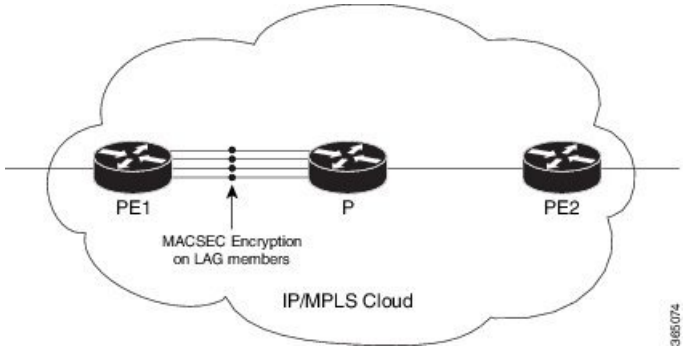
Figure 10: MACsec on Physical Interfaces in an MPLS Core Network



When MACsec is configured on the members of a LAG, an MKA session is set up for each member. SAK is exchanged for each LAG member and encryption or decryption takes place independently of other members in the group.

The following figure illustrates the use of MACsec on a link bundle in an MPLS core network.

Figure 11: MACsec on a Link Bundle in an MPLS Core Network



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

Create 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 *infinite*.

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.
- From Cisco IOS XR Software Release 7.2.1 and later, the MACsec key IDs (configured through CLI using the **macsec key** command under the key chain configuration mode) are considered to be case insensitive. These key IDs are stored as uppercase letters. For example, a key ID of value 'FF' and of value 'ff' are considered to be the same, and both these key IDs are now stored in uppercase as 'FF'. Whereas, prior to Release 7.2.1, both these values were treated as case sensitive, and hence considered as two separate key IDs. Hence it is recommended to have unique strings as key IDs for a MACsec key chain to avoid flapping of MACsec sessions. However, the support for this case insensitive IDs is applicable only for the configurations done through CLI, and not for configurations done through Netconf protocol.

Also, it is recommended to do a prior check of the MACsec key IDs before upgrading to Release 7.2.1 or later.

Consider a scenario where two MACsec key IDs with the same set of characters (say, ff and FF) are configured under the same key chain.

```
key chain 1
 macsec
  key ff
   lifetime 02:01:01 may 18 2020 infinite
  !
  key FF
   lifetime 01:01:01 may 18 2020 infinite
```

When you upgrade to Release 7.2.1 or later, only one of these key IDs is retained. That is 'FF', the one that was applied second in this example.

Follow these steps to configure a MACsec keychain:

Keychain Name: Provide a name for the MACsec keychain.

```
Router#configure
Router(config)#key chain kc
```

MACsec Mode: Enter the MACsec mode.

```
Router(config-kc)#macsec
Router(config-kc-MacSec)#
```

MACsec key: Provide a name for the MACsec key.

The MACsec 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.

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.

See the guidelines section to know more about the need for a unique key ID for a MACsec key chain.

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.

```
Router(config-kc-MacSec)#key 1234
```

AES Encryption: Enter the key string and the cryptographic algorithm to be used for the key.

```
! AES 128-bit encryption

Router(config-kc-MacSec-1234)#key-string 12345678
Router(config-kc-MacSec-1234)#cryptographic-algorithm AES-128-CMAC-96
```

MACsec key (CKN) lifetime or validity period: The lifetime period can be configured as a validity period between two dates (for example, Jan 01 2019 to Dec 31 2019), or with infinite validity. The key is valid from the time you configure (in HH:MM:SS format). Duration is configured in seconds.

Note When a key has expired, the MACsec session is torn down and running the **show macsec mka session** command does not display any information. If you run the **show macsec mka interface detail** command, the output displays ***** No Active Keys Present ***** in the PSK information.

```
Router(config-kc-MacSec-1234)#lifetime 05:00:00 01 January 2019 duration 1800
! Configuring  lifetime for a defined period

Router(config-kc-MacSec-1234)#lifetime 05:00:00 20 february 2019 12:00:00 30 september 2019


! Configuring  lifetime as infinite

Router(config-kc-MacSec-1234)#lifetime 05:00:00 01 January 2019 infinite
Router(config-kc-MacSec-1234)#commit
```

Securing the MACsec Pre-shared Key (PSK) Using Type 6 Password Encryption

Using the Type 6 password encryption feature, you can securely store MACsec plain text key string (CAK) in Type 6 encrypted format.

The primary key is the password or key used to encrypt all plain text MACsec key strings (CAK) in the router configuration with the use of an Advance Encryption Standard (AES) symmetric cipher. The primary key is not stored in the router configuration and cannot be seen or obtained in any way while connected to the router.

The Type 6 password encryption is effective only if a primary key is configured. The Type 6 Password Encryption is currently available on Cisco 8000 Series Routers.

Configuring a Primary Key and Enabling the Type 6 Password Encryption Feature

You can configure a primary key for Type 6 encryption and enable the Advanced Encryption Standard (AES) password encryption feature for securing the MACsec keys (key string/CAK). When prompted, enter the primary key details. The primary key can contain between 6 and 64 alphanumeric characters.

Primary Key Creation

```
Router#key config-key password-encryption
New password Requirements: Min-length 6, Max-length 64
Characters restricted to [A-Z][a-z][0-9]
Enter new key :
Enter confirm key :
```

Type 6 password encryption

```
Router#configure terminal
Router(config)#password6 encryption aes
Router(config)#commit
```

- **Modifying the Primary Key** - If a primary key is already configured, you are prompted to enter the current primary key before entering a new primary key.

  Modifying a primary key would re-encrypt all the existing Type 6 format key strings with the new primary key. If Type 6 key strings are present, ensure that the **password6 configuration aes** command is present to enable re-encryption with the new primary key. Otherwise, the primary key update operation fails.

- **Deleting the Primary Key** - Follow these steps to delete the primary key at any time.

```
Router# configure terminal
Router(config)#no password6 encryption aes
Router(config)#commit
Router(config)#exit
Router# key config-key password-encryption delete
```

Note Primary key deletion will bring down MACsec traffic if MKA sessions are up with Type 6 keys. To avoid traffic disruptions, configure a new set of PSK key pairs [key (CKN) and key string (CAK)] with latest timestamps with the lifetime of *infinite* validity on both the peers and ensure the successful CAK rekey to the newly configured CKN and CAK.

Configuring MACsec Pre-shared Key (PSK) For Type 6 Password Encryption

Ensure that you have configured a primary key using the **key config-key password-encryption** command and enabled the Type 6 encryption feature using the **password6 encryption aes** command.

```
Router#configure terminal
Router(config)#key chain kc1 macsec
Router(config-kc1-MacSec)# key 1111

! Configuring 32 byte hex CAK

Router(config-kc1-MacSec-1111)# key-string 12345678901234567890123456789022
cryptographic-algorithm aes-128-cmac

! Configuring 64 byte hex CAK

Router(config-kc1-MacSec-1111)# key-string
1234567890123456789012345678902212345678901234567890123456789022 cryptographic-algorithm
aes-256-cmac
Router(config-kc1-MacSec-1111)# lifetime 00:00:00 1 October 2019 infinite
Router(config-kc1-MacSec-1111)# commit
```

Running Configuration

The following is a sample output of the **show running-config key chain** command. The MACsec key chain name, the Type 6 key, and key-string information are displayed.

```
Router#show running-config key chain kc1

key chain kc1
 macsec
  key 1111
    key-string password6 674c434d695b5c5b464f546854474d58504946535455644b5043685969454861685
645464c5047635c464b5a6847566751535f45475266635256645a4a49454b4d6467514545434652546849576 5f6149674c5e4d
```

```
516843484446575c49685f5648665a604b5450435952634c61455d4b5e414142 cryptographic-algorithm
aes-256-cmac
        lifetime 00:00:00 october 01 2019 infinite
  !
```

Create a user-defined MACsec policy

- **MACsec policy** - Create a MACsec policy and configure the cipher suite to be used for MACsec encryption, including the confidentiality offset.

Note We recommend to change the offset value of the **conf-offset** <*offset_value*> command (MACsec encryption command) in the routers 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.

In this example, the GCM-AES-XPN-256 encryption algorithm is used. 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.

Note For Cisco 8000 Series Routers to interoperate with Cisco ASR9000 Series Routers that are older than Release 6.2.3, configure a user defined MACsec policy with the `policy-exception lacp-in-clear` command to bring up the MKA sessions over bundle interfaces running in LACP modes.

```
Router# configure terminal
Router(config)# macsec-policy mp-SF
Router(config-macsec-policy)# cipher-suite GCM-AES-XPN-128
Router(config-macsec-policy)# conf-offset CONF-OFFSET-30
```

- **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.

```
Router(config-macsec-policy)# key-server-priority 10
```

- **Security policy parameters** - Enable the Must-Secure or Should-Secure parameter.

  Must-Secure imposes that only MACsec encrypted traffic can flow. Hence, until the MKA session is secured, traffic is dropped.

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

Note Unlike in P2P scenarios, the traffic drops in P2MP if a third peer that is added with **should-secure** security policy fails to establish secure connection with two other peers that have already established a secure connection using **should-secure**. This is because, the first two peers update their configurations to drop all untagged (unencrypted) packets, once they establish a secure connection.

```
Router(config-macsec-policy)# security-policy should-secure
```

- **Data delay protection** - 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. When enabled, the **show macsec mka session interface** *interface* **detail** command output displays the **Delay Protection** field as **TRUE**.

  ```
  Router(config-macsec-policy)# delay-protection
  ```

- **Replay protection window size** - The window size dictates the maximum out-of-sequence frames that are accepted. You can configure a value between 0 and 1024.

- **ICV** for the frame arriving on the port - 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.

  ```
  Router(config-macsec-policy)# window-size 64
  Router(config-macsec-policy)# include-icv-indicator
  Router(config-macsec-policy)# commit
  ```

Running Configuration

The following is a sample output of the **show running-config macsec-policy** command.

```
Router# show running-config macsec-policy mp-SF

macsec-policy mp-SF
 conf-offset CONF-OFFSET-30
 security-policy should-secure
 cipher-suite GCM-AES-XPN-128
 window-size 64
 include-icv-indicator
 delay-protection
 key-server-priority 10
!
```

MACsec SAK Rekey Interval

You can set a timer value to rekey the MACsec secure association key (SAK) at a specified interval. This periodic refresh of SAK ensures that data encryption key is frequently updated. The configuration is effective on the node acting as a key server.

To set the rekey interval, use the **sak-rekey-interval** command in macsec-policy configuration mode. The timer ranges from 60 to 2,592,000 seconds, the default being OFF.

Configuration Example

```
Router#configure
Router(config)#macsec-policy test-policy
Router(config-macsec-policy)#sak-rekey-interval 120
Router(config-macsec-policy)#commit
```

Running Configuration

```
macsec-policy test-policy
 sak-rekey-interval 120
 !
```

Associated Command

sak-rekey-interval

EAPoL Ether-type and destination address

In WAN MACsec, when two peers establish an MKA session using the standard EAPoL Ether-type (0x888E) and destination MAC address (01:80:C2:00:00:03) through the service provider network, the Layer 2 intermediate devices may intercept and consume the EAPoL packets, which in turn can affect the MACsec session establishment between the two endpoints.

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
Alternate EAPoL Ether-type and destination address	Release 7.5.2	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200])(select variants only*). You can now configure an alternate EAPoL Ether-type, destination MAC address, or both under the MACsec-enabled interface on the following hardware. *This feature is now supported on the Cisco 8202-32FH-M routers.

Feature Name	Release Information	Feature Description
Alternate EAPoL Ether-type and destination address	Release 7.3.15	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q100, Q200])(select variants only*). Starting from this release, you can now configure an alternate EAPoL Ether-type, destination MAC address, or both under the MACsec-enabled interface on the following hardware. *This feature is now supported on the Cisco 88-LC0-36FH-M line cards.
Alternate EAPoL Ether-type and destination address	Release 7.0.12	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q100])(select variants only*). This feature allows to configure an alternate EAPoL Ether-type, destination MAC address, or both under the MACsec-enabled interface on the following hardware. *This feature is now supported on the Cisco 8800-LC-48H line cards.

The supported alternate EAPoL Ether-type is 0x876F. To configure an alternate EAPoL Ether-type, see Configure EAPoL Ether-type 0x876F.

The supported alternate EAPoL destination MAC address is the broadcast address FF:FF:FF:FF:FF:FF or the nearest bridge group address. To configure an alternate EAPoL destination address, see Configure EAPoL destination broadcast address and Configure EAPoL destination bridge group address.

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

- 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.
- To enable MACsec on bundles:
  - Enable MACsec on all 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.

Tip You can programmatically view the MACsec configuration using the `openconfig-macsec.yang` OpenConfig data model. To get started with using data models, see *Programmability Configuration Guide for Cisco 8000 Series Routers*.

MACsec PSK Configuration on an Interface

```
Router#configure
Router(config)#interface hundredGigE 0/1/0/10
Router(config-if)# macsec psk-keychain kc policy mp-SF
```

To enable MACsec PSK on a physical interface without the MACsec policy, use this command:

```
Router(config-if)#macsec psk-keychain script_kc
```

MACsec Fallback PSK Configuration on an Interface

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.

```
Router(config-if)#macsec psk-keychain kc fallback-psk-keychain fallback_kc policy mp-SF
Router(config-if)#commit
```

Verification

Before the introduction of active fallback functionality:

```
Router# show macsec mka session detail

NODE: node0_1_CPU0

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

Local Tx-SCI                    : 7872.5d1a.e7d4/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 7872.5d1a.e7d4
MKA Port Identifier             : 1
Interface Name                  : Hu0/1/0/10
CAK Name (CKN)                  : 1234
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : kc
Member Identifier (MI)          : C12A70FEE1212B835BDDDCBA
Message Number (MN)             : 395
Authenticator                   : NO
Key Server                      : NO
```

```
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-128

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 0
Latest SAK KI (KN)              : 018E2F0D63FF2ED6A5BF270E00000001 (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
Time to exit suspension         : NA

MKA Policy Name                 : mp-SF
Key Server Priority             : 16
Delay Protection                : FALSE
Replay Window Size              : 64
Include ICV Indicator           : FALSE
Confidentiality Offset          : 30
Algorithm Agility               : 80C201
SAK Cipher Suite                : 0080C20001000003 (GCM-AES-XPN-128)
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          SSCI  KS-Priority
-------------------------------------------------------------------------------
018E2F0D63FF2ED6A5BF270E       86       008a.962d.7400/0001    2        16

Potential Peer List:
-------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
-------------------------------------------------------------------------------

Peers Status:
 Last Tx MKPDU          : 2019 Oct 08 07:39:56.905
 Peer Count             : 1

 RxSCI                  : 008A962D74000001
  MI                    : 018E2F0D63FF2ED6A5BF270E
  Peer CAK              : Match
  Latest Rx MKPDU       : 2019 Oct 08 07:39:57.363
```

With the introduction of active fallback functionality:

The following is a snippet from the sample output that displays entry for active fallback PSK as well. The secured primary session output part is truncated here, which is exactly same as the output given above.

```
Router# show macsec mka session detail
NODE: node0_1_CPU0

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

Local Tx-SCI                    : 0257.3fae.5cda/0001
Local Tx-SSCI                   : 1
- - - - - - - - - - - - - - - - -
```

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

Local Tx-SCI                    : 0257.3fae.5cda/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 0257.3fae.5cda
MKA Port Identifier             : 1
Interface Name                  : Hu0/1/0/0
CAK Name (CKN)                  : 1111
CA Authentication Mode          : FALLBACK-PSK
Keychain                        : fb1
Member Identifier (MI)          : FC53A31E030E385981E0AACE
Message Number (MN)             : 178
Authenticator                   : NO
Key Server                      : NO
MKA Cipher Suite                : AES-256-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-256
Key Distribution Mode           : SAK

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 1
Latest SAK KI (KN)              : 725FF8F6605A3D428972538F00000001 (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
Time to exit suspension         : 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 : 0

Live Peer List:
--------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
--------------------------------------------------------------------------------
6A894FE1E984AD5314F33D21      188       0201.9ab0.85af/0001    0        16

Potential Peer List:
--------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
--------------------------------------------------------------------------------

Peers Status:
 Last Tx MKPDU          : 2021 Apr 30 14:56:33.797
 Peer Count             : 1

 RxSCI                  : 02019AB085AF0001
```

```
  MI                    : 6A894FE1E984AD5314F33D21
  Peer CAK              : Match
  Latest Rx MKPDU       : 2021 Apr 30 14:56:34.638
```

The following is a sample output of the **show macsec mka session interface** command. With this command, you can 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.

Verify whether the MKA session is secured with MACsec on the respective interface. 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.

```
Router#show macsec mka session interface hundredGigE 0/1/0/10

================================================================================================

   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN

================================================================================================

     Hu0/1/0/10      7872.5d1a.e7d4/0001     1      Secured      NO       PRIMARY     1234

     Hu0/1/0/10      7872.5d1a.e7d4/0001     1      Secured      NO       FALLBACK
5678
```

Before the introduction of active fallback functionality:

```
Router# show macsec mka session interface hundredGigE 0/1/0/10 detail

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

Local Tx-SCI                    : 7872.5d1a.e7d4/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 7872.5d1a.e7d4
MKA Port Identifier             : 1
Interface Name                  : Hu0/1/0/10
CAK Name (CKN)                  : 1234
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : kc
Member Identifier (MI)          : C12A70FEE1212B835BDDDCBA
Message Number (MN)             : 433
Authenticator                   : NO
Key Server                      : NO
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-128

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 0
Latest SAK KI (KN)              : 018E2F0D63FF2ED6A5BF270E00000001 (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
Time to exit suspension         : NA

MKA Policy Name                 : mp-SF
Key Server Priority             : 16
```

```
Delay Protection               : FALSE
Replay Window Size             : 64
Include ICV Indicator          : FALSE
Confidentiality Offset         : 30
Algorithm Agility              : 80C201
SAK Cipher Suite               : 0080C20001000003 (GCM-AES-XPN-128)
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          SSCI  KS-Priority
------------------------------------------------------------------------------
018E2F0D63FF2ED6A5BF270E      123       008a.962d.7400/0001    2        16

Potential Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------

Peers Status:
 Last Tx MKPDU          : 2019 Oct 08 07:41:12.929
 Peer Count             : 1

 RxSCI                  : 008A962D74000001
  MI                    : 018E2F0D63FF2ED6A5BF270E
  Peer CAK              : Match
  Latest Rx MKPDU       : 2019 Oct 08 07:41:11.400
```

With the introduction of active fallback functionality:

The following is a snippet from the sample output that displays entry for active fallback PSK as well. The secured primary session output part is truncated here, which is exactly same as the output given above.

```
Router# show macsec mka session interface hundredGigE 0/1/0/10 detail

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

Local Tx-SCI                   : 7872.5d1a.e7d4/0001
Local Tx-SSCI                  : 1
- - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - -
MKA Detailed Status for MKA Session
===================================
Status: Active - Marked Peer as Live (Waiting for SAK generation/distribution)

Local Tx-SCI                   : 34ed.1b5b.d0d7/0001
Local Tx-SSCI                  : 1
Interface MAC Address          : 34ed.1b5b.d0d7
MKA Port Identifier            : 1
Interface Name                 : Hu0/4/0/27
CAK Name (CKN)                 : 2222
CA Authentication Mode         : FALLBACK-PSK
Keychain                       : fb1
Member Identifier (MI)         : C0978A6B0916C3FC959773FE
Message Number (MN)            : 24039
Authenticator                  : NO
Key Server                     : NO
```

```
MKA Cipher Suite                : AES-256-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-256

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 2
Latest SAK KI (KN)              : 3D008A7D75DF0A9A35F9E3A900000002 (2)
Old SAK Status                  : No Rx, No Tx
Old SAK AN                      : 1
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
Time to exit suspension         : NA

MKA Policy Name                 : r1
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 : 0

Live Peer List:
-----------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
-----------------------------------------------------------------------------
B5ED6849883F34FEE89F74D1     26068      008a.9681.c02c/0001    2        16

Potential Peer List:
-----------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
-----------------------------------------------------------------------------

Peers Status:
Last Tx MKPDU          : 2021 Apr 28 02:08:03.795
Peer Count             : 1

RxSCI                  : 008A9681C02C0001
  MI                    : B5ED6849883F34FEE89F74D1
  Peer CAK              : Match
  Latest Rx MKPDU       : 2021 Apr 28 02:08:02.749
```

Configure EAPoL Ether-type 0x876F

This procedure allows to configure the EAPoL Ether-type 0x876F.

Procedure

Step 1 Create a MACsec key chain. For information, see Create a MACsec keychain.

Step 2 (Optional) Create a MACsec policy. For information, see Create a user-defined MACsec policy.

Step 3 Configure the EAPoL ether-type.

Example:

```
Router(config)# interface HundredGigE0/1/0/2
Router(config-if)# eapol eth-type 876F
Router(config-if)# commit
```

Step 4 Apply MACsec on a interface.

Example:

```
Router(config)# interface HundredGigE0/1/0/2
Router(config-if)# macsec psk-keychain kc fallback-psk-keychain fb
Router(config-if)# commit
```

Running Configuration

This example shows the running configuration for the EAPoL Ether-type 0x876F.

```
Router# show running-config interface HundredGigE0/1/0/2
interface HundredGigE0/1/0/2
  eapol eth-type 876F
  macsec psk-keychain kc fallback-psk-keychain fb
!
```

Verification

This example provides the verification for the EAPoL Ether-type 0x876F.

```
Router# show macsec mka interface  HundredGigE0/1/0/2 detail | i Ethertype
Ethertype                : 876F

Router# show macsec mka session interface HundredGigE0/1/0/2.1
===============================================================================================

   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN

===============================================================================================

     Hu0/1/0/2        0201.9ab0.77cd/0001     1      Secured      YES      PRIMARY     1234

     Hu0/1/0/2        0201.9ab0.77cd/0001     1      Active       YES     FALLBACK     9999
```

Configure EAPoL destination broadcast address

This procedure allows to configure the EAPoL destination address with the broadcast address FF:FF:FF:FF:FF, to ensure that the underlying L2 network floods the EAPoL packets to all receivers.

Procedure

Step 1 Create a MACsec key chain. For information, see Create a MACsec keychain.

Step 2 (Optional) Create a MACsec policy. For information, see Create a user-defined MACsec policy.

Step 3 Configure the EAPoL destination address.

Example:

```
Router(config)# interface HundredGigE0/1/0/2
Router(config-if)# eapol destination-address broadcast-address
Router(config-if)# commit
```

Step 4 Apply MACsec on a interface.

Example:

```
Router(config)# interface HundredGigE0/1/0/2
Router(config-if)# macsec psk-keychain kc fallback-psk-keychain fb
Router(config-if)# commit
```

Running Configuration

This example shows the running configuration for the EAPoL destination broadcast address.

```
Router# show running-config interface HundredGigE0/1/0/2
 eapol destination-address ffff.ffff.ffff
 macsec psk-keychain kc fallback-psk-keychain fb
!
```

Verification

This example provides the verification for the EAPoL destination broadcast address.

```
Router# show macsec mka interface  HundredGigE0/1/0/2 detail  | i EAPoL
    EAPoL Destination Addr   : ffff.ffff.ffff

Router# show macsec mka session interface HundredGigE0/1/0/2
================================================================================================

   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN

================================================================================================

     Hu0/1/0/2       02df.3638.d568/0001     1      Secured      YES      PRIMARY     1234

     Hu0/1/0/2       02df.3638.d568/0001     1      Active       YES     FALLBACK     9999
```

Configure EAPoL destination bridge group address

This procedure allows to set the EAPoL destination address with the nearest bridge group address, for example 01:80:C2:00:00:00 and provides the EAPoL destination address configuration on a physical interface, which is inherited by the MACsec-enabled subinterface.

Procedure

Step 1 Create a MACsec key chain. For information, see Create a MACsec keychain.

Step 2 (Optional) Create a MACsec policy. For information, see Create a user-defined MACsec policy.

Step 3 Configure the EAPoL destination bridge group address on a MACsec-enabled physical interface.

Example:

```
Router(config)# interface HundredGigE0/1/0/1
Router(config-if)# eapol destination-address bridge-group-address
Router(config-if)# commit
```

Step 4 Configure MACsec on a subinterface.

Example:

```
Router(config)# interface HundredGigE0/1/0/1.1
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# macsec psk-keychain kc fallback-psk-keychain fb
outer(config-subif)# commit
```

Running Configuration

This example shows the running configuration for the EAPoL destination bridge group address on the MACsec-enabled physical interface.

```
Router# show running-config interface Hu0/1/0/1
interface HundredGigE0/1/0/1
eapol destination-address 0180.c200.0000
```

This example shows the running configuration for the EAPoL destination bridge group address on the MACsec-enabled subinterface.

```
Router# show running-config interface HundredGigE0/1/0/1.1
interface HundredGigE0/1/0/0.1
  macsec psk-keychain kc fallback-psk-keychain fb
  encapsulation dot1q 1
!
```

Verification

This example provides the verification for the EAPoL destination bridge group address configured on the MACsec-enabled subinterface.

```
Router# show macsec mka interface  HundredGigE0/1/0/1.1 detail  | i EAPoL
    EAPoL Destination Addr   : 0180.c200.0000

Router# show macsec mka session interface HundredGigE0/1/0/1.1
===============================================================================================

   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN

===============================================================================================

     Hu0/1/0/1.1       0201.9ab0.85af/0001     1      Secured      YES      PRIMARY
1234
     Hu0/1/0/1.1       0201.9ab0.85af/0001     1      Active       YES     FALLBACK
9999
```

MACsec mode on PHY

MACsec mode is a mode that ensures allocation of the required power for all MACsec ports on a line card or a router. Enabling MACsec mode prevents interface flap when MACsec feature is enabled on a port.

Enable MACsec Mode on PHY

You can enable the MACsec mode for the PHY of a line card or of a fixed router by using the **hw-module macsec-mode** command in XR Config mode mode.

Configuration Example

```
Router#configure
Router(config)#hw-module macsec-mode location 0/1/CPU0
Router(config)#commit
```

Note You must reload the line card for the MACsec mode configuration to take effect.

Note If the MACsec mode is already enabled on a node such as a line card, then the system does not allow you to configure the **hw-module macsec-mode location all** command again. This restriction is in place to prevent conflicts in configuration, especially in a configuration restore scenario. In such scenarios, you can make use of the **show hw-module macsec-mode** command to know of the respective running configurations in place.

Running Configuration

```
hw-module macsec-mode location 0/1/CPU0
!
```

Verification

You can use the **show hw-module macsec-mode** command in the XR EXEC mode to display the MACsec mode of line cards, and the user action to be performed.

```
Router#show hw-module macsec-mode location 0/1/CPU0
Sat Dec  7 14:31:52.668 UTC
Location       Configured     Running          Action
--------------------------------------------------------------
0/1/CPU0       YES            NO               RELOAD
```

You can also use the **show hw-module macsec-mode location all** command to display the MACsec mode information of all nodes. This **location all** option is available starting Cisco IOS XR Software Release 7.0.14.

```
Router#show hw-module macsec-mode location all
Sun Feb 16 21:06:07.726 UTC

Location       Configured     Running          Action
--------------------------------------------------------------
0/0/CPU0       YES            NO               RELOAD
0/7/CPU0       NO             NO               NONE
```

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.

From Cisco IOS XR Software Release 7.3.1 and later, an alternative option, **allow**, is introduced under the macsec-policy configuration mode, that allows packets to be sent in clear-text format. You can use the **allow lacp-in-clear** command for LACP packets.

Similarly, you can use the **allow pause-frames-in-clear** to allow Ethernet PAUSE frame packets in clear text, from Release 7.3.15 and later.

How to Create MACsec Policy Exception

Note The **policy-exception lacp-in-clear** command under macsec-policy configuration mode is deprecated. Hence, it is recommended to use the **allow lacp-in-clear** command instead, to allow LACP packets in clear-text format.

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
```

Using the **allow** command:

```
Router#configure
Router(config)#macsec-policy test-macsec-policy
Router(config-macsec-policy)#allow lacp-in-clear
Router(config-macsec-policy)#allow pause-frames-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
!
```

With the **allow** command:

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

```
  allow lacp-in-clear
  allow pause-frames-in-clear
  security-policy should-secure
  include-icv-indicator
  sak-rekey-interval seconds 120
!
```

Associated Commands

- **policy-exception lacp-in-clear**
- **allow lacp-in-clear**
- **allow pause-frames-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.

Note With the introduction of active fallback functionality in Cisco IOS XR Software Release 7.0.14, the output of various MACsec show commands include the fallback PSK entry as well.

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

SUMMARY STEPS

1. Verify the MACsec policy configuration.
2. Verify the MACsec configuration on the respective interface.
3. Verify whether the interface of the router is peering with its neighbor after MACsec configuration.
4. Verify whether the MKA session is secured with MACsec on the respective interface.
5. Verify the MACsec session counter statistics.

DETAILED STEPS

Procedure

Step 1 Verify the MACsec policy configuration.

Example:

```
Router# show macsec policy mp-SF

================================================================================
Policy            Cipher              Key-Svr   Window    Conf      Delay
name              Suite               Priority  Size      Offset    Protection
================================================================================
mp-SF             GCM-AES-XPN-128     16        64        30        FALSE
```

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) or P2MP interface (VPLS network).

Example:

```
Router# show macsec mka summary

NODE: node0_1_CPU0
================================================================================================
   Interface-Name     Status     Cipher-Suite       KeyChain       PSK/EAP       CKN
================================================================================================
     Hu0/1/0/10       Secured  GCM-AES-XPN-128         kc          PRIMARY     1234

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

With the introduction of active fallback functionality:

The following is a sample output that displays active fallback PSK entry as well:

```
Router# show macsec mka summary
Wed Apr 28 01:50:57.543 UTC

NODE: node0_4_CPU0
================================================================================================
   Interface-Name     Status     Cipher-Suite       KeyChain       PSK/EAP       CKN
================================================================================================
     Hu0/4/0/27       Secured  GCM-AES-XPN-256         kc1          PRIMARY     1111
     Hu0/4/0/27       Active   GCM-AES-XPN-256         fb1         FALLBACK     2222

NODE: node0_6_CPU0
================================================================================================
   Interface-Name     Status     Cipher-Suite       KeyChain       PSK/EAP       CKN
================================================================================================
     Hu0/6/0/29       Secured  GCM-AES-XPN-256         kc1          PRIMARY     1111
     Hu0/6/0/29       Active   GCM-AES-XPN-256         fb1         FALLBACK     2222

Total MACSec Sessions  : 4
    Secured Sessions   : 2
    Pending Sessions   : 0
    Suspended Sessions : 0
      Active  Sessions : 2
```

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. The **#Peers** field in the output confirms the number of peers you have configured on the physical interface. If the number of peers is not reflected accurately in this output, run the **show run** command and verify the peer configuration on the interface.

Example:

```
Router#show macsec mka session

NODE: node0_1_CPU0
================================================================================================
   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN
================================================================================================
     Hu0/1/0/10      7872.5d1a.e7d4/0001     1      Secured      NO        PRIMARY     1234
```

The following is a sample output that displays active fallback PSK entry as well:

```
Router# show macsec mka session
Wed Apr 28 01:59:39.478 UTC

NODE: node0_4_CPU0
================================================================================================
   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN
================================================================================================
     Hu0/4/0/27      34ed.1b5b.d0d7/0001     1      Secured      NO        PRIMARY     1111
     Hu0/4/0/27      34ed.1b5b.d0d7/0001     1      Active       NO       FALLBACK     2222

NODE: node0_6_CPU0
================================================================================================
   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN
================================================================================================
     Hu0/6/0/29      00d6.fee7.fe44/0001     1      Secured      YES       PRIMARY     1111
     Hu0/6/0/29      00d6.fee7.fe44/0001     1      Active       YES      FALLBACK     2222
```

Note
In the VPLS network, because of the configuration on a multipoint interface, the number of live peers displayed is more than 1.

```
Router#show macsec mka session
Mon Nov 23 11:20:39.835 UTC

NODE: node0_4_CPU0
================================================================
Interface-Name Local-TxSCI #Peers Status Key-Server PSK/EAP CKN
================================================================
FH0/4/0/34 34ed.1b5b.d10f/0001 2 Secured YES PRIMARY  1111
FH0/4/0/34 34ed.1b5b.d10f/0001 2 Active  YES FALLBACK 2222
Hu0/4/0/28 34ed.1b5b.d0df/0001 2 Secured NO PRIMARY   1111
Hu0/4/0/28 34ed.1b5b.d0df/0001 2 Active  NO FALLBACK  2222
Hu0/4/0/29 34ed.1b5b.d0e7/0001 2 Secured NO PRIMARY   1111
Hu0/4/0/29 34ed.1b5b.d0e7/0001 2 Active  NO FALLBACK  2222
```

Before the introduction of active fallback functionality:

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

```
Router# show macsec mka session detail

NODE: node0_1_CPU0

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

Local Tx-SCI                    : 7872.5d1a.e7d4/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 7872.5d1a.e7d4
MKA Port Identifier             : 1
Interface Name                  : Hu0/1/0/10
```

```
CAK Name (CKN)                  : 1234
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : kc
Member Identifier (MI)          : C12A70FEE1212B835BDDDCBA
Message Number (MN)             : 3009
Authenticator                   : NO
Key Server                      : NO
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-128

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 0
Latest SAK KI (KN)              : 018E2F0D63FF2ED6A5BF270E00000001 (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
Time to exit suspension         : NA

MKA Policy Name                 : mp-SF
Key Server Priority             : 16
Delay Protection                : FALSE
Replay Window Size              : 64
Include ICV Indicator           : FALSE
Confidentiality Offset          : 30
Algorithm Agility               : 80C201
SAK Cipher Suite                : 0080C20001000003 (GCM-AES-XPN-128)
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          SSCI  KS-Priority
------------------------------------------------------------------------------
018E2F0D63FF2ED6A5BF270E      2699      008a.962d.7400/0001    2        16

Potential Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------

Peers Status:
 Last Tx MKPDU          : 2019 Oct 08 09:07:06.475
 Peer Count             : 1

 RxSCI                  : 008A962D74000001
  MI                    : 018E2F0D63FF2ED6A5BF270E
  Peer CAK              : Match
  Latest Rx MKPDU       : 2019 Oct 08 09:07:06.032
```

With the introduction of active fallback functionality:

The following is a snippet from the sample output that displays entry for active fallback PSK as well. The secured primary session output part is truncated here, which is exactly same as the output given above.

```
Router# show macsec mka session detail
NODE: node0_1_CPU0
```

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

Local Tx-SCI                    : 0257.3fae.5cda/0001
Local Tx-SSCI                   : 1
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -

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

Local Tx-SCI                    : 0257.3fae.5cda/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 0257.3fae.5cda
MKA Port Identifier             : 1
Interface Name                  : Hu0/1/0/0
CAK Name (CKN)                  : 1111
CA Authentication Mode          : FALLBACK-PSK
Keychain                        : fb1
Member Identifier (MI)          : FC53A31E030E385981E0AACE
Message Number (MN)             : 178
Authenticator                   : NO
Key Server                      : NO
MKA Cipher Suite                : AES-256-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-256
Key Distribution Mode           : SAK

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 1
Latest SAK KI (KN)              : 725FF8F6605A3D428972538F00000001 (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
Time to exit suspension         : 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 : 0

Live Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------
6A894FE1E984AD5314F33D21      188       0201.9ab0.85af/0001    0        16

Potential Peer List:
------------------------------------------------------------------------------
```

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

Peers Status:
 Last Tx MKPDU          : 2021 Apr 30 14:56:33.797
 Peer Count             : 1

 RxSCI                  : 02019AB085AF0001
  MI                    : 6A894FE1E984AD5314F33D21
  Peer CAK              : Match
  Latest Rx MKPDU       : 2021 Apr 30 14:56:34.638
```

Note
If the MKA session status is shown as **Secured** with **0 (Zero)** peer count, this means that the link is locally secured (Tx). This is because of MKA peer loss caused by **No Rx Packets (MKA Packet)** from that peer.

Step 4 Verify whether the MKA session is secured with MACsec on the respective interface.

Example:

```
Router# show macsec mka session interface hundredGigE 0/1/0/10

=============================================================================================
   Interface-Name        Local-TxSCI       #Peers   Status   Key-Server   PSK/EAP      CKN
=============================================================================================
     Hu0/1/0/10      7872.5d1a.e7d4/0001     1      Secured      NO       PRIMARY     1234
     Hu0/1/0/10      7872.5d1a.e7d4/0001     1      Secured      NO       FALLBACK     5678
```

Before the introduction of active fallback functionality:

```
Router# show macsec mka session interface hundredGigE 0/1/0/10 detail

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

Local Tx-SCI                    : 7872.5d1a.e7d4/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 7872.5d1a.e7d4
MKA Port Identifier             : 1
Interface Name                  : Hu0/1/0/10
CAK Name (CKN)                  : 1234
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : kc
Member Identifier (MI)          : C12A70FEE1212B835BDDDCBA
Message Number (MN)             : 3058
Authenticator                   : NO
Key Server                      : NO
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-128

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 0
Latest SAK KI (KN)              : 018E2F0D63FF2ED6A5BF270E00000001 (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
Time to exit suspension         : NA

MKA Policy Name                 : mp-SF
```

```
Key Server Priority             : 16
Delay Protection                : FALSE
Replay Window Size              : 64
Include ICV Indicator           : FALSE
Confidentiality Offset          : 30
Algorithm Agility               : 80C201
SAK Cipher Suite                : 0080C20001000003 (GCM-AES-XPN-128)
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          SSCI  KS-Priority
--------------------------------------------------------------------------------
018E2F0D63FF2ED6A5BF270E      2748      008a.962d.7400/0001    2        16

Potential Peer List:
--------------------------------------------------------------------------------
           MI                   MN             Rx-SCI          SSCI  KS-Priority
--------------------------------------------------------------------------------

Peers Status:
 Last Tx MKPDU          : 2019 Oct 08 09:08:44.506
 Peer Count             : 1

 RxSCI                  : 008A962D74000001
  MI                    : 018E2F0D63FF2ED6A5BF270E
  Peer CAK              : Match
  Latest Rx MKPDU       : 2019 Oct 08 09:08:44.081
```

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 **INITIALIZING**.

With the introduction of active fallback functionality:

The following is a snippet from the sample output that displays entry for active fallback PSK as well. The secured primary session output part is truncated here, which is exactly same as the output given above.

```
Router# show macsec mka session interface hundredGigE 0/1/0/10 detail

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

Local Tx-SCI                    : 7872.5d1a.e7d4/0001
Local Tx-SSCI                   : 1
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
MKA Detailed Status for MKA Session
===================================
Status: Active - Marked Peer as Live (Waiting for SAK generation/distribution)

Local Tx-SCI                    : 34ed.1b5b.d0d7/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : 34ed.1b5b.d0d7
MKA Port Identifier             : 1
Interface Name                  : Hu0/4/0/27
CAK Name (CKN)                  : 2222
CA Authentication Mode          : FALLBACK-PSK
```

```
Keychain                          : fb1
Member Identifier (MI)            : C0978A6B0916C3FC959773FE
Message Number (MN)               : 24039
Authenticator                     : NO
Key Server                        : NO
MKA Cipher Suite                  : AES-256-CMAC
Configured MACSec Cipher Suite    : GCM-AES-XPN-256

Latest SAK Status                 : Rx & Tx
Latest SAK AN                     : 2
Latest SAK KI (KN)                : 3D008A7D75DF0A9A35F9E3A900000002 (2)
Old SAK Status                    : No Rx, No Tx
Old SAK AN                        : 1
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
Time to exit suspension           : NA

MKA Policy Name                   : r1
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 : 0

Live Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------
B5ED6849883F34FEE89F74D1     26068      008a.9681.c02c/0001    2        16

Potential Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------

Peers Status:
Last Tx MKPDU          : 2021 Apr 28 02:08:03.795
Peer Count             : 1

RxSCI                  : 008A9681C02C0001
  MI                    : B5ED6849883F34FEE89F74D1
  Peer CAK              : Match
  Latest Rx MKPDU       : 2021 Apr 28 02:08:02.749
```

In a VPLS network with multipoint interface, the output would display more than one peer as follows:

```
Router#show macsec mka session interface Hu0/3/0/16 detail
Thu Oct 29 10:09:25.586 UTC


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

```
Local Tx-SCI                    : fc58.9a05.9aa0/0001
Local Tx-SSCI                   : 4
Interface MAC Address           : fc58.9a05.9aa0
MKA Port Identifier             : 1
Interface Name                  : Hu0/3/0/16
CAK Name (CKN)                  : 1234
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : kc
Member Identifier (MI)          : C45D5F2028232022F30C6BD8
Message Number (MN)             : 9427
Authenticator                   : NO
Key Server                      : YES
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-256

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 1
Latest SAK KI (KN)              : C45D5F2028232022F30C6BD800000036 (54)
Old SAK Status                  : No Rx, No Tx
Old SAK AN                      : 0
Old SAK KI (KN)                 : RETIRED (53)

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
Time to exit suspension         : NA

MKA Policy Name                 : ms
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           : 3
# of MACsec Capable Live Peers Responded : 3

# of MACSec Suspended Peers              : 0

Live Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------
499BFCA3044D65A9BA4FC219      9427      fc58.9a05.9aa8/0001    3        16
56765F41D6BE434860E62991      9427      fc58.9a05.9ab0/0001    2        16
7236084C87ADD66D59C63FE1      9425      fc58.9a05.9ab1/0001    1        16

Potential Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------

Suspended Peer List:
------------------------------------------------------------------------------
       Rx-SCI          SSCI
------------------------------------------------------------------------------

Peers Status:
 Last Tx MKPDU          : 2020 Oct 29 10:09:25.071
 Peer Count             : 3
```

```
 RxSCI                  : FC589A059AB00001
  MI                    : 56765F41D6BE434860E62991
  Peer CAK              : Match
  Latest Rx MKPDU       : 2020 Oct 29 10:09:24.072

 RxSCI                  : FC589A059AA80001
  MI                    : 499BFCA3044D65A9BA4FC219
  Peer CAK              : Match
  Latest Rx MKPDU       : 2020 Oct 29 10:09:25.574

 RxSCI                  : FC589A059AB10001
  MI                    : 7236084C87ADD66D59C63FE1
  Peer CAK              : Match
  Latest Rx MKPDU       : 2020 Oct 29 10:09:24.572
```

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:

```
Router# show macsec mka statistics interface hundredGigE 0/1/0/10

MKA Statistics for Session on interface (Hu0/1/0/10)
=====================================================
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.......... 0
   SAKs Rekeyed............ 0
   SAKs Received........... 1
   SAK Responses Received.. 0

MKPDU Statistics
   MKPDUs Transmitted...... 3097
      "Distributed SAK".. 0
      "Distributed CAK".. 0
   MKPDUs Validated & Rx... 2788
      "Distributed SAK".. 1
      "Distributed CAK".. 0

MKA IDB Statistics
   MKPDUs Tx Success.......... 3097
   MKPDUs Tx Fail............. 0
   MKPDUS Tx Pkt build fail... 0
   MKPDUS No Tx on intf down.. 3
   MKPDUS No Rx on intf down.. 0
   MKPDUs Rx CA Not found..... 0
   MKPDUs Rx Error............ 0
   MKPDUs Rx Success.......... 2788
   MKPDUs Rx Invalid Length... 0
   MKPDUs Rx Invalid CKN...... 0
   MKPDUs Rx force suspended.. 0
   MKPDUs Tx force suspended.. 0

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
   MKPDU Rx Drop Packet, Ethertype Mismatch.. 0
   MKPDU Rx Drop Packet, Source MAC NULL..... 0
   MKPDU Rx Drop Packet, Destination MAC NULL 0
   MKPDU Rx Drop Packet, Payload NULL........ 0

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

CA Failures
   ICK Derivation.................... 0
   KEK Derivation.................... 0
   Invalid Peer MACsec Capability... 0

MACsec Failures
   Rx SC Creation.................... 0
   Tx SC Creation.................... 0
   Rx SA Installation................ 0
   Tx SA Installation................ 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 Cisco 8000 Series Routers

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.

SUMMARY STEPS

1. Verify the MACsec encryption and hardware interface descriptor block (IDB) information on the interface.
2. To verify if the hardware programming is done, use the following command:

DETAILED STEPS

Procedure

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

Example:

```
Router# show macsec ea idb interface hundredGigE 0/1/0/10

IDB Details:
  if_sname                : Hu0/1/0/10
  if_handle               : 0x8001e0
  MacSecControlledIfh     : 0x800330
  MacSecUnControlledIfh   : 0x800338
  Replay window size      : 64
  Local MAC               : 78:72:5d:1a:e7:d4
  Rx SC Option(s)         : Validate-Frames Replay-Protect
  Tx SC Option(s)         : Protect-Frames Always-Include-SCI
  Security Policy         : SHOULD SECURE
  Delay Protection        : FALSE
  Sectag offset           : 0
  Rx SC 1
    Rx SCI                : 008a962d74000001
    Peer MAC              : 00:8a:96:2d:74:00
    Stale                 : NO
    SAK Data
      SAK[0]              : ***
      SAK Len             : 16
      SAK Version         : 1
      HashKey[0]          : ***
      HashKey Len         : 16
      Conf offset         : 30
      Cipher Suite        : GCM-AES-XPN-128
      CtxSalt[0]          : 01 8f 2f 0f 63 ff 2e d6 a5 bf 27 0e
      ssci                : 2
      Rx SA Program Req[0]: 2019 Oct 08 07:37:14.870
      Rx SA Program Rsp[0]: 2019 Oct 08 07:37:14.902

  Tx SC
    Tx SCI                : 78725d1ae7d40001
    Active AN             : 0
    Old AN                : 255
    Next PN               : 1, 0, 0, 0
    SAK Data
      SAK[0]              : ***
      SAK Len             : 16
      SAK Version         : 1
      HashKey[0]          : ***
      HashKey Len         : 16
      Conf offset         : 30
      Cipher Suite        : GCM-AES-XPN-128
      CtxSalt[0]          : 01 8f 2f 0c 63 ff 2e d6 a5 bf 27 0e
      ssci                : 1
      Tx SA Program Req[0]: 2019 Oct 08 07:37:14.908
      Tx SA Program Rsp[0]: 2019 Oct 08 07:37:14.931
```

When more than 1 RX SA is configured in P2MP networks, the output would be as follows:

```
Router#show macsec ea idb interface Hu0/3/0/16
Thu Oct 29 10:10:10.947 UTC

IDB Details:
  if_sname                : Hu0/3/0/16
  if_handle               : 0x1800240
  MacSecControlledIfh     : 0x1800270
  MacSecUnControlledIfh   : 0x1800278
  Replay window size      : 64
  Local MAC               : fc:58:9a:05:9a:a0
  Rx SC Option(s)         : Validate-Frames Replay-Protect
  Tx SC Option(s)         : Protect-Frames Always-Include-SCI
  Security Policy         : MUST SECURE
```

```
Delay Protection        : FALSE
Sectag offset           : 0
Rx SC 1
  Rx SCI                : fc589a059ab10001
  Peer MAC              : fc:58:9a:05:9a:b1
  Stale                 : NO
  SAK Data
    SAK[1]              : ***
    SAK Len             : 32
    SAK Version         : 3
    HashKey[1]          : ***
    HashKey Len         : 16
    Conf offset         : 0
    Cipher Suite        : GCM-AES-XPN-256
    CtxSalt[1]          : c4 6b 5f 21 28 23 20 22 f3 0c 6b d8
    ssci                : 1
    Rx SA Program Req[1]: 2020 Oct 29 05:04:30.803
    Rx SA Program Rsp[1]: 2020 Oct 29 05:04:30.807

Rx SC 2
  Rx SCI                : fc589a059ab00001
  Peer MAC              : fc:58:9a:05:9a:b0
  Stale                 : NO
  SAK Data
    SAK[1]              : ***
    SAK Len             : 32
    SAK Version         : 3
    HashKey[1]          : ***
    HashKey Len         : 16
    Conf offset         : 0
    Cipher Suite        : GCM-AES-XPN-256
    CtxSalt[1]          : c4 6b 5f 22 28 23 20 22 f3 0c 6b d8
    ssci                : 2
    Rx SA Program Req[1]: 2020 Oct 29 05:04:30.792
    Rx SA Program Rsp[1]: 2020 Oct 29 05:04:30.796

Rx SC 3
  Rx SCI                : fc589a059aa80001
  Peer MAC              : fc:58:9a:05:9a:a8
  Stale                 : NO
  SAK Data
    SAK[1]              : ***
    SAK Len             : 32
    SAK Version         : 3
    HashKey[1]          : ***
    HashKey Len         : 16
    Conf offset         : 0
    Cipher Suite        : GCM-AES-XPN-256
    CtxSalt[1]          : c4 6b 5f 23 28 23 20 22 f3 0c 6b d8
    ssci                : 3
    Rx SA Program Req[1]: 2020 Oct 29 05:04:30.788
    Rx SA Program Rsp[1]: 2020 Oct 29 05:04:30.792

Tx SC
  Tx SCI                : fc589a059aa00001
  Active AN             : 1
  Old AN                : 0
  Next PN               : 1, 1, 1, 1
  SAK Data
    SAK[1]              : ***
    SAK Len             : 32
    SAK Version         : 3
    HashKey[1]          : ***
    HashKey Len         : 16
```

```
    Conf offset         : 0
    Cipher Suite        : GCM-AES-XPN-256
    CtxSalt[1]          : c4 6b 5f 24 28 23 20 22 f3 0c 6b d8
    ssci                : 4
    Tx SA Program Req[1]: 2020 Oct 29 05:04:32.773
    Tx SA Program Rsp[1]: 2020 Oct 29 05:04:32.780
```

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 To verify if the hardware programming is done, use the following command:

Example:

```
Router# show macsec platform hardware sa interface hundredGigE 0/1/0/10
----------------------------------------------------------------
Tx SA Details:
----------------------------------------------------------------
    SCI                   : 7872.5d1a.e7d4/0001
    Crypto Algo           : GCM-AES-XPN-128
    AES Key Len           : 128 bits
    AN                    : 0
    Initial Packet Number : 1
    Current Packet Number : 1
    Maximum Packet Number : 3221225400
    XForm in Use          : YES
    Action Type           : SA Action Egress
    Direction             : Egress
    Conf Offset           : 00000030
    Drop Type             : 0x00000003
    SA In Use             : YES
    ConfProtect           : YES
    IncludeSCI            : YES
    ProtectFrame          : YES
    UseEs                 : NO
    UseSCB                : NO

----------------------------------------------------------------
Rx SA Details:
----------------------------------------------------------------
    SCI                   : 008a.962d.7400/0001
    Replay Window         : 64
    Crypto Algo           : GCM-AES-XPN-128
    AES Key Len           : 128 bits
    AN                    : 0
    Initial Packet Number : 1
    Current Packet Number : 1
    Maximum Packet Number : 3221225400
    XForm in Use          : YES
    Action Type           : SA Action Ingress
    Direction             : Ingress
    Conf Offset           : 00000030
    Drop Type             : 0x00000002
    SA In Use             : YES
    ReplayProtect         : YES
```

```
lowPN                : 1
nxtPN                : 2
```

This completes the verification of MACsec encryption on the router hardware, and the configuration and verification of MACsec encryption.

MACsec SecY Statistics

The following methods are used to query MACsec SecY statistics such as, encryption, decryption, and the hardware statistics.

- CLI
- SNMP MIB

Querying SNMP Statistics Using CLI

The following example shows how to query SNMP statistics using a CLI. Use the **show macsec secy statistics interface** *interface name* command to display the MACsec SecY statistics details.

Note When you use the **show macsec secy statistics interface** command in the 8712-MOD-M routers, all TxSC counters will display value of zero. This is due to a hardware limitation, as the collection of TxSC statistics is not supported in K100 ASIC-based systems like the Cisco 8712-MOD-M routers.

```
Router# show macsec secy stats interface hundredGigE 0/1/0/10 sc

Interface Stats
    InPktsUntagged     :  0
    InPktsNoTag        :  0
    InPktsBadTag       :  0
    InPktsUnknownSCI   :  0
    InPktsNoSCI        :  0
    InPktsOverrun      :  0
    InOctetsValidated  :  0
    InOctetsDecrypted  :  0
    OutPktsUntagged    :  0
    OutPktsTooLong     :  0
    OutOctetsProtected :  0
    OutOctetsEncrypted :  0

SC Stats
  TxSC Stats
    OutPktsProtected   : 0
    OutPktsEncrypted   : 0
    OutOctetsProtected : 0
    OutOctetsEncrypted : 0
    OutPktsTooLong     : 0
    TxSA Stats
      TxSA 0:
        OutPktsProtected : 0
        OutPktsEncrypted : 0
        NextPN           : 1
```

```
    TxSA 1:
      OutPktsProtected : 0
      OutPktsEncrypted : 0
      NextPN           : 0
    TxSA 2:
      OutPktsProtected : 0
      OutPktsEncrypted : 0
      NextPN           : 0
    TxSA 3:
      OutPktsProtected : 0
      OutPktsEncrypted : 0
      NextPN           : 0

RxSC Stats
  RxSC 1: 10000742d968a00
    InPktsUnchecked     : 0
    InPktsDelayed       : 0
    InPktsLate          : 0
    InPktsOK            : 0
    InPktsInvalid       : 0
    InPktsNotValid      : 0
    InPktsNotUsingSA    : 0
    InPktsUnusedSA      : 0
    InPktsUntaggedHit   : 0
    InOctetsValidated   : 0
    InOctetsDecrypted   : 0
  RxSA Stats
    RxSA 0:
      InPktsUnusedSA      : 0
      InPktsNotUsingSA    : 0
      InPktsNotValid      : 0
      InPktsInvalid       : 0
      InPktsOK            : 0
      NextPN              : 1
    RxSA 1:
      InPktsUnusedSA      : 0
      InPktsNotUsingSA    : 0
      InPktsNotValid      : 0
      InPktsInvalid       : 0
      InPktsOK            : 0
      NextPN              : 0
    RxSA 2:
      InPktsUnusedSA      : 0
      InPktsNotUsingSA    : 0
      InPktsNotValid      : 0
      InPktsInvalid       : 0
      InPktsOK            : 0
      NextPN              : 0
    RxSA 3:
      InPktsUnusedSA      : 0
      InPktsNotUsingSA    : 0
      InPktsNotValid      : 0
      InPktsInvalid       : 0
      InPktsOK            : 0
      NextPN              : 0
```

MACsec SNMP MIB (IEEE8021-SECY-MIB)

The IEEE8021-SECY-MIB provides Simple Network Management Protocol (SNMP) access to the MAC security entity (SecY) MIB running with IOS XR MACsec-enabled line cards. The IEEE8021-SECY-MIB is used to query on the SecY data, encryption, decryption, and the hardware statistics. The SecY MIB data is queried only on the Controlled Port.

The object ID of the IEEE8021-SECY-MIB is 1.0.8802.1.1.3. The IEEE8021-SECY-MIB contains the following tables that specifies the detailed attributes of the MACsec Controlled Port interface index.

Table 21: IEEE8021-SECY-MIB Table

Tables	OID
secyIfTable	1.0.8802.1.1.3.1.1.1
secyTxSCTable	1.0.8802.1.1.3.1.1.2
secyTxSATable	1.0.8802.1.1.3.1.1.3
secyRxSCTable	1.0.8802.1.1.3.1.1.4
secyRxSATable	1.0.8802.1.1.3.1.1.5
secyCipherSuiteTable	1.0.8802.1.1.3.1.1.6
secyTxSAStatsTable	1.0.8802.1.1.3.1.2.1
secyTxSCStatsTable	1.0.8802.1.1.3.1.2.2
secyRxSAStatsTable	1.0.8802.1.1.3.1.2.3
secyRxSCStatsTable	1.0.8802.1.1.3.1.2.4
secyStatsTable	1.0.8802.1.1.3.1.2.5

For more information, see the SecY IEEE MIB at the following URL:

http://www.ieee802.org/1/files/public/MIBs/IEEE8021-SECY-MIB-200601100000Z.mib

secyIfTable

The following table represents the system level information for each interface supported by the MAC security entity. The index tuple for this table is secyIfInterfaceIndex.

Table 22: secyIfTable

Object	Object identifier
secyIfInterfaceIndex	1.0.8802.1.1.3.1.1.1.1.1
secyIfMaxPeerSCs	1.0.8802.1.1.3.1.1.1.1.2
secyIfRxMaxKeys	1.0.8802.1.1.3.1.1.1.1.3
secyIfTxMaxKeys	1.0.8802.1.1.3.1.1.1.1.4
secyIfProtectFramesEnable	1.0.8802.1.1.3.1.1.1.1.5
secyIfValidateFrames	1.0.8802.1.1.3.1.1.1.1.6
secyIfReplayProtectEnable	1.0.8802.1.1.3.1.1.1.1.7

Object	Object identifier
secyIfReplayProtectWindow	1.0.8802.1.1.3.1.1.1.1.8
secyIfCurrentCipherSuite	1.0.8802.1.1.3.1.1.1.1.9
secyIfAdminPt2PtMAC	1.0.8802.1.1.3.1.1.1.1.10
secyIfOperPt2PtMAC	1.0.8802.1.1.3.1.1.1.1.11
secyIfIncludeSCIEnable	1.0.8802.1.1.3.1.1.1.1.12
secyIfUseESEnable	1.0.8802.1.1.3.1.1.1.1.13
secyIfUseSCBEnable	1.0.8802.1.1.3.1.1.1.1.14

secyTxSCTable

The following table provides information about the status of each transmitting SC supported by the MAC security entity. The index tuple for this table is secyIfInterfaceIndex.

Table 23: secyTxSCTable

Object	Object identifier
secyTxSCI	1.0.8802.1.1.3.1.1.2.1.1
secyTxSCState	1.0.8802.1.1.3.1.1.2.1.2
secyTxSCEncodingSA	1.0.8802.1.1.3.1.1.2.1.3
secyTxSCEncipheringSA	1.0.8802.1.1.3.1.1.2.1.4
secyTxSCCreatedTime	1.0.8802.1.1.3.1.1.2.1.5
secyTxSCStartedTime	1.0.8802.1.1.3.1.1.2.1.6
secyTxSCStoppedTime	1.0.8802.1.1.3.1.1.2.1.7

secyTxSATable

The following table provides information about the status of each transmitting SA supported by the MAC security entity. The index tuple for this table is: {secyIfInterfaceIndex, secyTxSA}.

Table 24: secyTxSATable

Object	Object identifier
secyTxSA	1.0.8802.1.1.3.1.1.3.1.1
secyTxSAState	1.0.8802.1.1.3.1.1.3.1.2
secyTxSANextPN	1.0.8802.1.1.3.1.1.3.1.3

Object	Object identifier
secyTxSAConfidentiality	1.0.8802.1.1.3.1.1.3.1.4
secyTxSASAKUnchanged	1.0.8802.1.1.3.1.1.3.1.5
secyTxSACreatedTime	1.0.8802.1.1.3.1.1.3.1.6
secyTxSAStartedTime	1.0.8802.1.1.3.1.1.3.1.7
secyTxSAStoppedTime	1.0.8802.1.1.3.1.1.3.1.8

secyRxSCTable

The following table provides information about the status of each receiving SC supported by the MAC security entity. The index tuple for this table is: {secyIfInterfaceIndex, secyRxSCI}.

Table 25: secyRxSCTable

Object	Object identifier
secyRxSCI	1.0.8802.1.1.3.1.1.4.1.1
secyRxSCState	1.0.8802.1.1.3.1.1.4.1.2
secyRxSCCurrentSA	1.0.8802.1.1.3.1.1.4.1.3
secyRxSCCreatedTime	1.0.8802.1.1.3.1.1.4.1.4
secyRxSCStartedTime	1.0.8802.1.1.3.1.1.4.1.5
secyRxSCStoppedTime	1.0.8802.1.1.3.1.1.4.1.6

secyRxSATable

The following table provides information about the status of each receiving SA supported by the MAC security entity. The index tuple for this table is: {secyIfInterfaceIndex, secyRxSCI, secyRxSA}.

Table 26: secyRxSATable

Object	Object identifier
secyRxSA	1.0.8802.1.1.3.1.1.5.1.1
secyRxSAState	1.0.8802.1.1.3.1.1.5.1.2
secyRxSANextPN	1.0.8802.1.1.3.1.1.5.1.3
secyRxSASAKUnchanged	1.0.8802.1.1.3.1.1.5.1.4
secyRxSACreatedTime	1.0.8802.1.1.3.1.1.5.1.5
secyRxSAStartedTime	1.0.8802.1.1.3.1.1.5.1.6
secyRxSAStoppedTime	1.0.8802.1.1.3.1.1.5.1.7

secyCipherSuiteTable

The following table is a list of selectable cipher suites for the MAC security entity. The index tuple for this table is: {secyCipherSuiteIndex}.

Table 27: secyCipherSuiteTable

Object	Object identifier
secyCipherSuiteIndex	1.0.8802.1.1.3.1.1.6.1.1
secyCipherSuiteId	1.0.8802.1.1.3.1.1.6.1.2
secyCipherSuiteName	1.0.8802.1.1.3.1.1.6.1.3
secyCipherSuiteCapability	1.0.8802.1.1.3.1.1.6.1.4
secyCipherSuiteProtection	1.0.8802.1.1.3.1.1.6.1.5
secyCipherSuiteProtectionOffset	1.0.8802.1.1.3.1.1.6.1.6
secyCipherSuiteDataLengthChange	1.0.8802.1.1.3.1.1.6.1.7
secyCipherSuiteICVLength	1.0.8802.1.1.3.1.1.6.1.8
secyCipherSuiteRowStatus	1.0.8802.1.1.3.1.1.6.1.9

secyTxSAStatsTable

The following table that contains the statistics objects for each transmitting SA in the MAC security entity.

Table 28: secyTxSAStatsTable

Object	Object identifier
secyTxSAStatsProtectedPkts	1.0.8802.1.1.3.1.2.1.1.1
secyTxSAStatsEncryptedPkts	1.0.8802.1.1.3.1.2.1.1.2
secyTxSCStatsProtectedPkts	1.0.8802.1.1.3.1.2.2.1.1
secyTxSCStatsEncryptedPkts	1.0.8802.1.1.3.1.2.2.1.4
secyTxSCStatsOctetsProtected	1.0.8802.1.1.3.1.2.2.1.10
secyTxSCStatsOctetsEncrypted	1.0.8802.1.1.3.1.2.2.1.11

secyTxSCStatsTable

The following table that contains the statistics objects for each transmitting SC in the MAC security entity.

Table 29: secyTxSCStatsTable

Object	Object identifier
secyTxSCStatsProtectedPkts	1.0.8802.1.1.3.1.2.2.1.1
secyTxSCStatsEncryptedPkts	1.0.8802.1.1.3.1.2.2.1.4
secyTxSCStatsOctetsProtected	1.0.8802.1.1.3.1.2.2.1.10
secyTxSCStatsOctetsEncrypted	1.0.8802.1.1.3.1.2.2.1.11

secyRxSAStatsTable

The following table that contains the statistics objects for each receiving SA in the MAC security entity.

Table 30: secyRxSAStatsTable

Object	Object identifier
secyRxSAStatsUnusedSAPkts	1.0.8802.1.1.3.1.2.3.1.1
secyRxSAStatsNoUsingSAPkts	1.0.8802.1.1.3.1.2.3.1.4
secyRxSAStatsNotValidPkts	1.0.8802.1.1.3.1.2.3.1.13
secyRxSAStatsInvalidPkts	1.0.8802.1.1.3.1.2.3.1.16
secyRxSAStatsOKPkts	1.0.8802.1.1.3.1.2.3.1.25

secyRxSCStatsTable

The following table that contains the statistics objects for each receiving SC in the MAC security entity.

Table 31: secyRxSCStatsTable

Object	Object identifier
secyRxSCStatsUnusedSAPkts	1.0.8802.1.1.3.1.2.4.1.1
secyRxSCStatsNoUsingSAPkts	1.0.8802.1.1.3.1.2.4.1.2
secyRxSCStatsLatePkts	1.0.8802.1.1.3.1.2.4.1.3
secyRxSCStatsNotValidPkts	1.0.8802.1.1.3.1.2.4.1.4
secyRxSCStatsInvalidPkts	1.0.8802.1.1.3.1.2.4.1.5
secyRxSCStatsDelayedPkts	1.0.8802.1.1.3.1.2.4.1.6
secyRxSCStatsUncheckedPkts	1.0.8802.1.1.3.1.2.4.1.7
secyRxSCStatsOKPkts	1.0.8802.1.1.3.1.2.4.1.8
secyRxSCStatsOctetsValidated	1.0.8802.1.1.3.1.2.4.1.9

Object	Object identifier
secyRxSCStatsOctetsDecrypted	1.0.8802.1.1.3.1.2.4.1.10

secyStatsTable

The following table lists the objects for the statistics information of each Secy supported by the MAC security entity.

Table 32: secyStatsTable

Object	Object identifier
secyStatsTxUntaggedPkts	1.0.8802.1.1.3.1.2.5.1.1
secyStatsTxTooLongPkts	1.0.8802.1.1.3.1.2.5.1.2
secyStatsRxUntaggedPkts	1.0.8802.1.1.3.1.2.5.1.3
secyStatsRxNoTagPkts	1.0.8802.1.1.3.1.2.5.1.4
secyStatsRxBadTagPkts	1.0.8802.1.1.3.1.2.5.1.5
secyStatsRxUnknownSCIPkts	1.0.8802.1.1.3.1.2.5.1.6
secyStatsRxNoSCIPkts	1.0.8802.1.1.3.1.2.5.1.7
secyStatsRxOverrunPkts	1.0.8802.1.1.3.1.2.5.1.8

Obtaining the MACsec Controlled Port Interface Index

The ifindex of the controlled port can be obtained using the following commands:

- **snmpwalk** command on IfMib[OID: 1.3.6.1.2.1.31.1.1.1]

```
rtr1.0/1/CPU0/ $ snmpwalk -v2c -c public 10.0.0.1  1.3.6.1.2.1.31.1.1.1.1
SNMPv2-SMI::mib-2.31.1.1.1.1.3 = STRING: "GigabitEthernet0/1/0/0"
SNMPv2-SMI::mib-2.31.1.1.1.1.18 = STRING: "MACSecControlled0/1/0/0"
SNMPv2-SMI::mib-2.31.1.1.1.1.19 = STRING: "MACSecUncontrolled0/1/0/0"
```

- **show snmp interface** command

```
Router# show snmp interface
.
.
ifName : MACSecControlled0/0/0/0  ifIndex : 77
ifName : MACSecControlled0/0/0/4  ifIndex : 79
ifName : MACSecControlled0/0/0/21  ifIndex : 94
ifName : MACSecControlled0/0/0/30  ifIndex : 118
ifName : MACSecControlled0/0/0/34  ifIndex : 116
ifName : MACSecUncontrolled0/0/0/0  ifIndex : 78
ifName : MACSecUncontrolled0/0/0/4  ifIndex : 80
ifName : MACSecUncontrolled0/0/0/21  ifIndex : 95
ifName : MACSecUncontrolled0/0/0/30  ifIndex : 119
ifName : MACSecUncontrolled0/0/0/34  ifIndex : 117
```

SNMP Query Examples

In the following examples, it is assumed that the configured SNMP community is public, and the management IP of the box is 10.0.0.1.

To perform SNMP walk on the entire SECY MIB for the router, use the following command:

snmpwalk -v2c -c public 10.0.0.1 1.0.8802.1.1.3

To query on the secyTxSCTable to get the TxSCI for interface Gi0/1/0/0, using the ifindex of MACsecControlled0/1/0/0 that is 18, use the following command:

snmpget -v2c -c public 10.0.0.1 iso.0.8802.1.1.3.1.1.2.1.1.18

Power-on Self-Test KAT for Common Criteria and FIPS

The Cisco IOS XR Software Release 7.0.14 introduces the support for power-on self-test (POST) known answer test (KAT) for common criteria and FIPS compliance for the MACsec-enabled hardware on Cisco 8000 Series Routers. In POST KAT, the KAT is executed immediately after the cipher module is powered on. With this feature enabled, the system allows the traffic to pass through the MACsec-enabled hardware only if it passes the KAT. If the KAT fails, the modules shut down and the ports do not come up.

The POST KAT functionality is now available on Cisco 8800 48x100 GbE QSFP28 Line Card (8800-LC-48H) and Cisco 8800 36x400GE QSFP56-DD Line Card with MACsec (8800-LC-36FH-M).

How to Configure Power-on Self-Test KAT

KAT is not enabled by default. You can configure the **hw-module macsec-fips-post** command to enable POST KAT for the MACsec-enabled hardware. With this configuration in place, the KAT always runs as a self-test during power on. The cryptographic algorithm tests are performed on every physical layer chip (PHY) with hardware crypto once it powered up.

Note

- With KAT enabled, you can expect a delay of approximately 2 to 3 minutes for the boot up of a line card, in comparison to the boot up time without enabling KAT.
- If power-on self-test (POST) known answer test (KAT) is already enabled on the PHY, then the system does not allow you to configure the **hw-module macsec-fips-post location all** command again. This restriction is in place to prevent conflicts in configuration, especially in a configuration restore scenario. In such scenarios, you can make use of the **show hw-module macsec-mode fips-post** command to know of the respective running configurations in place.

Pass criteria for KAT: Any change in the FIPS mode configuration requires a line card reload. On reload, the FIPS POST is run as part of the LC boot sequence. The subsequent boot (based on the FIPS mode) state re-triggers the KAT. If there are multiple PHYs hardware in a module, then the system performs the KAT on each of the PHYs and returns the KAT results. If all PHYs pass the KAT, then the system brings up the line card for regular usage.

Fail criteria for KAT: Traffic does not pass through a MACsec-enabled PID that failed KAT. If any of the PHYs registers a KAT failure, then the module enters into an ERROR state and the system displays a critical ERROR SYSLOG output which reads as: *KAT Test Failed*. The system does not allow any traffic or data

flow through the interfaces on that line card. Although the interfaces are present, they do not come up or allow any traffic to flow through them on a line card that failed KAT. In a modular chassis, all other line cards, except the one that failed the KAT, will be up and running.

Prerequisites for Power-on Self-Test KAT

- The k9sec package must be installed on the router.
- FIPS must be supported and enabled on the line card.

Configuration Example

```
Router#config
Router(config)#hw-module macsec-fips-post location 0/4/CPU0
Router(config)#commit
```

Running Configuration

```
hw-module macsec-fips-post location 0/4/CPU0
!
```

Verification

Before configuring POST KAT:

```
Router#show hw-module macsec-fips-post
Wed Jun 17 09:29:18.780 UTC

Location       Configured     Applied          Action
--------------------------------------------------------------
0/0/CPU0       NO             NO               NONE          >>> LC36
0/11/CPU0      NO             NO               NONE          >>> LC48
```

After configuring the command for POST KAT, and before the line card reload:

```
Router#show hw-module macsec-fips-post
Wed Jun 17 09:36:31.932 UTC

Location       Configured     Applied          Action
--------------------------------------------------------------
0/0/CPU0       NO             NO               NONE
0/11/CPU0      YES            NO               RELOAD
```

After the line card reload:

```
Router#show hw-module macsec-fips-post
Wed Jun 17 10:03:57.263 UTC

Location       Configured     Applied          Action
--------------------------------------------------------------
0/0/CPU0       NO             NO               NONE
0/11/CPU0      YES            YES              NONE
```

These are sample logs displayed after a successful KAT. The system performs KAT on each port, but the ports may not be in order in the display output.

```
Router#show logging | inc KAT
Wed Jun 10 12:07:29.849 UTC
LC/0/4/CPU0:Jun 9 10:37:37.521 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 0
LC/0/4/CPU0:Jun 9 10:37:37.522 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 28
LC/0/4/CPU0:Jun 9 10:37:37.522 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 27
LC/0/4/CPU0:Jun 9 10:37:37.522 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 1
LC/0/4/CPU0:Jun 9 10:39:10.393 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 2
LC/0/4/CPU0:Jun 9 10:39:10.393 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 6
LC/0/4/CPU0:Jun 9 10:39:10.393 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 7
LC/0/4/CPU0:Jun 9 10:39:10.393 UTC: optics_driver[159]: %L2-SECY_DRIVER-6-KAT_PASS : KAT
Test PASSED for Port No: 8
```

These are sample logs displayed in KAT failure scenarios:

```
Router#show logging | inc SECY
Thu Jul 16 09:13:29.217 UTC
LC/0/7/CPU0:Jul 16 08:41:30.709 UTC: optics_driver[152]: %L2-SECY_DRIVER-0-KAT_FAIL_DETECTED
 : KAT Test FAILED for Port No: 0
LC/0/7/CPU0:Jul 16 08:41:30.709 UTC: optics_driver[152]: %L2-SECY_DRIVER-0-KAT_FAIL_DETECTED
 : KAT Test FAILED for Port No: 47
LC/0/7/CPU0:Jul 16 08:41:30.709 UTC: optics_driver[152]: %L2-SECY_DRIVER-0-KAT_FAIL_DETECTED
 : KAT Test FAILED for Port No: 7
LC/0/7/CPU0:Jul 16 08:41:30.709 UTC: optics_driver[152]: %L2-SECY_DRIVER-0-KAT_FAIL_DETECTED
 : KAT Test FAILED for Port No: 6
```

Related Topics

- Power-on Self-Test KAT for Common Criteria and FIPS, on page 207

Associated Commands

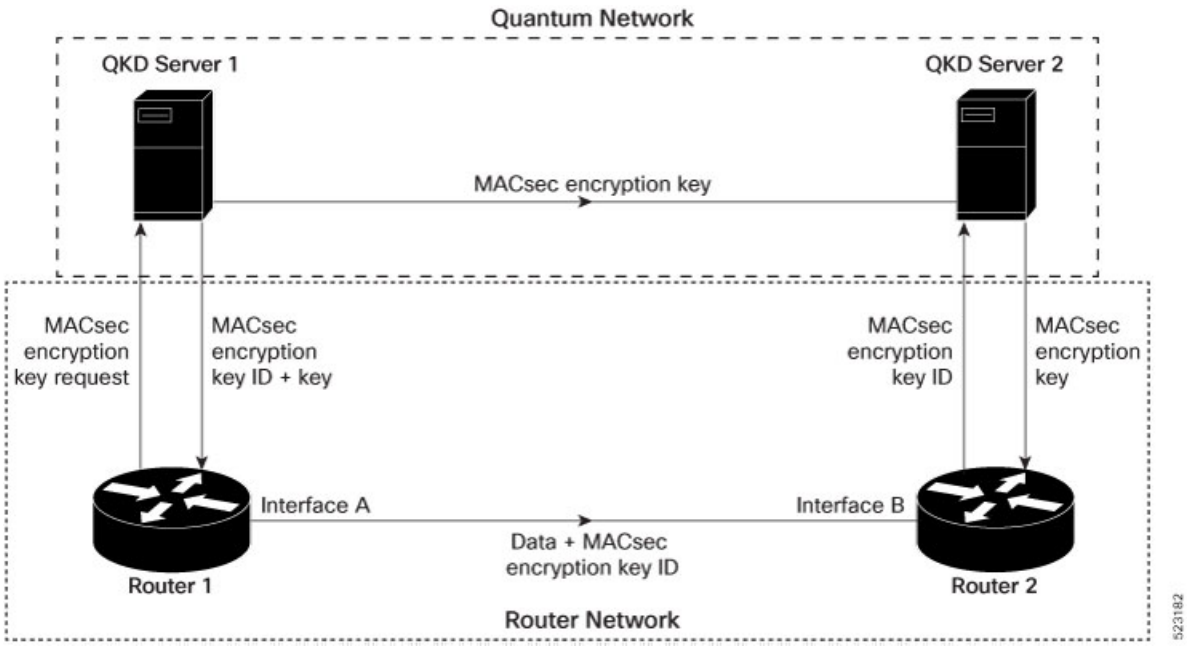
- **hw-module macsec-fips-post**
- **show hw-module macsec-fips-post**

Secure Key Integration Protocol

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
Secure Key Integration Protocol for Routers	Release 7.9.1	Your routers are now capable of handling the Secure Key Integration Protocol (SKIP) protocol. The SKIP protocol enables your routers to communicate with external quantum devices. With this ability, you can use the Quantum Key Distribution (QKD) devices for exchanging MACsec encryption keys between routers. Using QKD eliminates the key distribution problem in a post quantum world where the current cryptographic systems are no longer secure due to the advent of quantum computers. This feature introduces the following: • **CLI:** • **crypto-sks-kme** • **show crypto sks profile** • **Yang Data Model**: Cisco-IOS-XR-um-sks-server-cfg.yang (see GitHub, YANG Data Models Navigator) For more information on Quantum Key Distribution, see Post Quantum Security Brief.

Cisco Secure Key Integration Protocol (SKIP) enables your router that supports encryption to use keys by a quantum distribution system. SKIP implementation in Cisco IOS-XR software supports integrating external Quantum Key Distribution (QKD) devices with your routers. With integration support between the routers and QKD devices, you can use the QKD devices to exchange encryption keys for communication between the routers. And this mechanism eliminates the key distribution problem in a post quantum world.



Quantum Key Distribution (QKD) is a method for securely transmitting a secret key between two parties. QKD uses the laws of quantum mechanics to guarantee security even when eavesdroppers monitor the communication channel. In QKD, the key is encoded in the states of single photons. The QKD transmits the keys over optical fiber or free space (vacuum). The security of the key relies on the fact that measuring a quantum state introduces a change in the quantum state. The change in quantum states helps the two end parties of the communication channel to identify any interception of their key.

QKD is a secure key exchange mechanism against quantum attacks and will remain so, even with future advancements in cryptanalysis or quantum computing. Unlike other cryptographic algorithms, QKD doesn't need continual updates based on discovered vulnerabilities.

Feature Highlights

- You can use the QKD devices in the following combinations:
  - Same QKD device on the end ports of the peer routers
  - Different QKD devices on the end ports of the peer routers
  - Multiple links between the same peer routers using different QKD devices
- You can use a specific source interface for the router communication with the QKD devices. To use a specific source interface, configure the source interface in the QKD profile. Use the **source interface** command in SKS configuration mode as follows.

```
Router# config
Router(config)# sks profile ProfileR1toR2 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.34 port 10001
Router(config-sks-profile)# source interface hundredGigE 0/1/0/17
Router(config-sks-profile)# commit
```

- You can use an HTTP Proxy for the router communication with the QKD devices. Use the following configuration for the router to use an HTTP proxy server to communicate to the QKD devices.

```
Router# config
Router(config)# sks profile ProfileR1toR2 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.34 port 10001
Router(config-sks-profile)# http proxy ipv4 192.0.2.68 port 804
Router(config-sks-profile)# commit
```

Note The **http proxy server** command supports configuration using IPv4 address, IPv6 address, and hostname of the HTTP proxy.

Restrictions

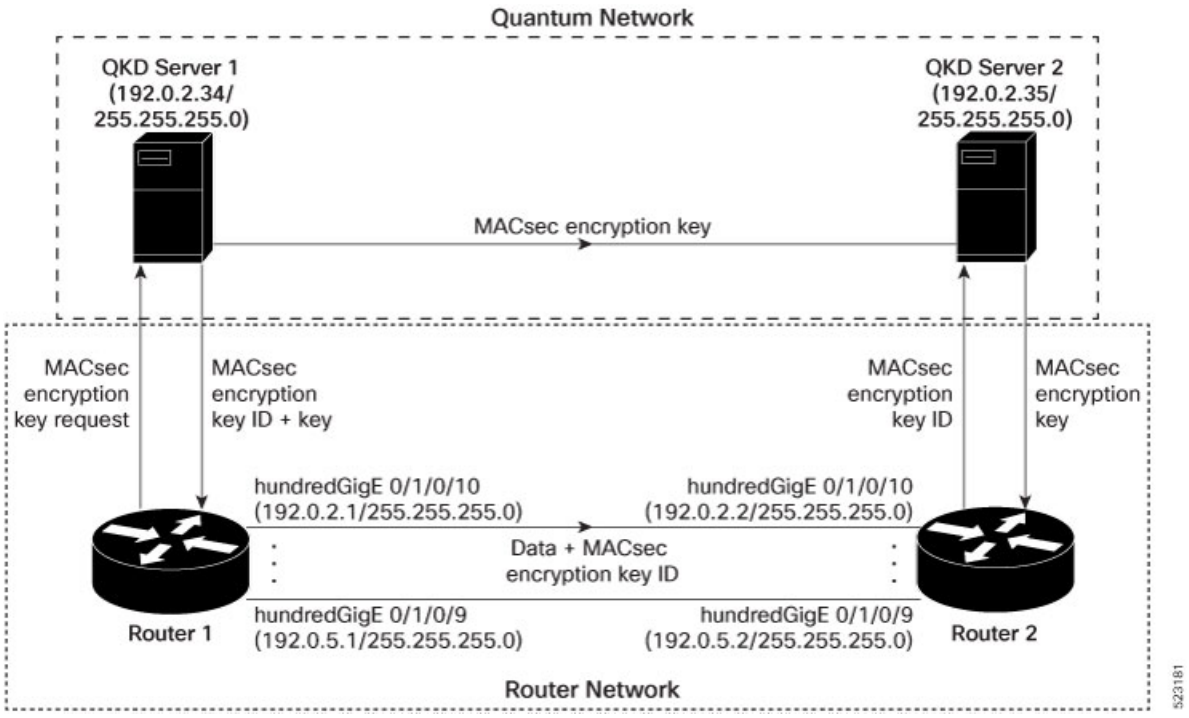
Consider the following restrictions before implementing SKIP:

- The SKIP protocol is supported only on the 8202-32FH-M chassis.
- You can use the SKIP protocol only in a Point to Point MACSec link encryption scenario.
- The SKIP protocol is available only on the interfaces that support MACSec encryption.

Configuring Point to Point MACsec Link Encryption using SKIP

In Point-to-Point MACsec Link Encryption, the router uses SKIP to establish secure encryption. This encryption is set up between two interfaces in peer routers and requires the assistance of an external QKD device network. The QKD network shares the MACsec encryption key instead of the router network. Thus, when the router needs to create a MACsec link between peer router interfaces, it contacts the external QKD device and requests the key. The external QKD device generates a Key pair comprising the Key ID and the Key. The Key ID serves as the unique identification string for the Key (Shared Secret). The QKD then shares both the Key ID and Key with the router and the router shares only the Key ID with its peer. The Peer router uses this Key ID to retrieve encryption keys from its QKD device. Therefore, Quantum networks securely communicate encryption keys always.

Figure 12: Point to Point MACsec Link Encryption using SKIP



Prerequisites

- Configure MACsec Pre-Sared Key (PSK). For more information, see MACsec PSK, on page 162.
- Configure MACsec in the PPK mode.
- An external QKD devices network.
- Add the QKD server CA to the trustpoint in the router. For more information, see Declare Certification Authority and Configure Trusted Point.
- Import the QKD server root CA certificate in the router. For more information, see Configure Certificate Enrollment Using Cut-and-Paste.

Configuration

The following example details how to establish Point to Point MACsec Link Encryption using SKIP:

Router 1:

1. Configure the QKD profile.

```
Router# config
Router(config)# sks profile ProfileR1toR2 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.34 port 10001
Router(config-sks-profile)# commit
```

2. Map the QKD profile to the MACsec policy.

```
Router# config
Router(config)# macsec-policy R1toR2
```

```
Router(config-macsec-policy)# ppk sks-profile ProfileR1toR2
Router(config-macsec-policy)# commit
```

Note For more information on MACsec Policy, see Create a user-defined MACsec policy, on page 170.

3. Apply MACsec policy to the interfaces.

```
Router# config
Router(config)#interface hundredGigE 0/1/0/10
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
Router(config)#interface hundredGigE 0/1/0/11
Router(config-if)# ipv4 address 192.0.3.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
Router(config)#interface hundredGigE 0/1/0/12
Router(config-if)# ipv4 address 192.0.4.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
Router(config)#interface hundredGigE 0/1/0/9
Router(config-if)# ipv4 address 192.0.5.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
```

Router 2:

1. Configure the QKD profile.

```
Router# config
Router(config)# sks profile ProfileR2toR1 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.35 port 10001
Router(config-sks-profile)# commit
```

2. Map the QKD profile to the MACsec policy.

```
Router# config
Router(config)# macsec-policy R2toR1
Router(config-macsec-policy)# ppk sks-profile ProfileR2toR1
Router(config-macsec-policy)# commit
```

Note For more information on MACsec Policy, see Create a user-defined MACsec policy, on page 170.

3. Apply MACsec policy to the interfaces.

```
Router# config
Router(config)#interface hundredGigE 0/1/0/10
Router(config-if)# ipv4 address 192.0.2.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
Router(config)#interface hundredGigE 0/1/0/11
Router(config-if)# ipv4 address 192.0.3.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
Router(config)#interface hundredGigE 0/1/0/12
Router(config-if)# ipv4 address 192.0.4.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
```

```
Router(config)#interface hundredGigE 0/1/0/9
Router(config-if)# ipv4 address 192.0.5.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
```

Running Configuration

Router 1:

```
sks profile ProfileR1toR2 type remote
 kme server ipv4 192.0.2.34 port 10001
!
macsec-policy R1toR2
 ppk
  sks-profile ProfileR1toR2
 !
!
interface hundredGigE 0/1/0/10
 ipv4 address 192.0.2.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
interface hundredGigE 0/1/0/11
 ipv4 address 192.0.3.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
interface hundredGigE 0/1/0/12
 ipv4 address 192.0.4.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
interface hundredGigE 0/1/0/9
 ipv4 address 192.0.5.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
```

Router 2:

```
sks profile ProfileR2toR1 type remote
 kme server ipv4 192.0.2.35 port 10001
!
macsec-policy R2toR1
 ppk
  sks-profile ProfileR2toR1
 !
!
interface hundredGigE 0/1/0/10
 ipv4 address 192.0.2.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!t
interface hundredGigE 0/1/0/11
 ipv4 address 192.0.3.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!
interface hundredGigE 0/1/0/12
 ipv4 address 192.0.4.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!
interface hundredGigE 0/1/0/9
 ipv4 address 192.0.5.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!
```

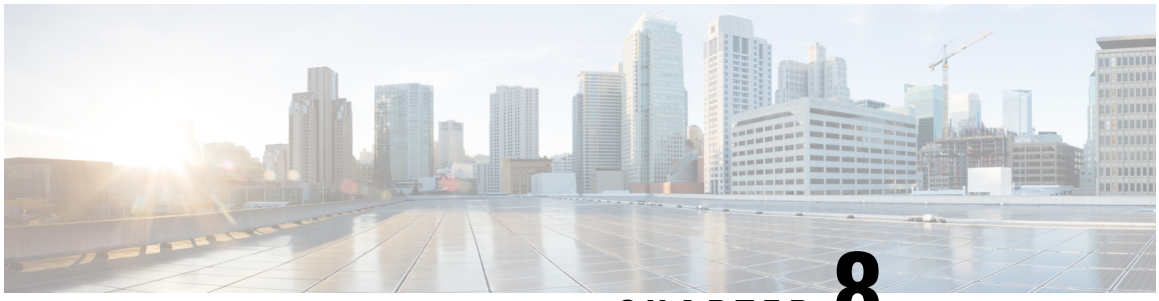
Verification

```
Router(ios)# show crypto sks profile all
Profile Name         :ProfileR1toR2
Myidentifier        :Router1
Type                :Remote
Reg Client Count     :1

Server
IP                   :192.0.2.34
Port                 :10001
Vrf                  :Notconfigured
Source Interface     :Notconfigured
Status               :Connected
Entropy              :true
Key                  :true
Algorithm            :QKD
Local identifier     :Alice
Remote identifier    :Alice

Peerlist
QKD ID               :Bob
State                :Connected

Peerlist
QKD ID               :Alice
State                :Connected


Router(ios)# show crypto sks profile all stats
Profile Name            : ProfileR1toR2
My identifier           : Router1
Server
 IP                     : 192.0.2.34
 Port                   : 10001
 Status                 : connected
Counters
 Capability request         : 1
 Key request                : 3
 Key-id request             : 0
 Entropy request            : 0
 Capability response        : 1
 Key response               : 3
 Key-id response            : 0
 Entropy response           : 0
 Total request              : 4
 Request failed             : 0
 Request success            : 4
 Total response             : 4
 Response failed            : 0
 Response success           : 4
 Retry count                : 0
 Response Ignored           : 0
 Cancelled count            : 0
Response time
 Max Time                   : 100 ms
 Avg Time                   : 10  ms
 Min Time                   : 50  ms
Last transaction
 Transaction Id             : 9
 Transaction type           : Get key
 Transaction status         : Response data received, successfully
 Http code                  : 200 OK (200)
```

Related Commands for MACsec

The following commands are available to verify the SNMP results.

Command	Description
show macsec mka session detail	Displays the details of all MACsec Key Agreement (MKA) sessions on the device.
show macsec mka interface detail	Verifies the MACsec MKA status on the interface.
show macsec ea idb interface	Verifies the MACsec encryption and hardware interface descriptor block (IDB) information on the interface.



CHAPTER 8

MACSec Using EAP-TLS Authentication

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.

- Guidelines and Limitations for EAP-TLS Authentication, on page 219
- IEEE 802.1X Device Roles, on page 220
- Prerequisites for MACSec MKA Using EAP-TLS Authentication, on page 220
- MACsec with Local EAP-TLS Authentication, on page 220
- Configure MACSec Encryption Using EAP-TLS Authentication, on page 220
- Configure RADIUS Server, on page 220
- Configure 802.1X Authentication Method, on page 221
- Generate RSA Key Pair, on page 221
- Configure Trustpoint, on page 222
- Configure Domain Name, on page 223
- Authenticate Certificate Authority and Request Certificates, on page 223
- Configure EAP Profile, on page 224
- Configure 802.1X Profile on the Device, on page 224
- Configure MACSec EAP and 802.1X Profile on an Interface, on page 225
- Verify MACSec EAP and 802.1X Configuration on Interface, on page 225

Guidelines and Limitations for EAP-TLS Authentication

The EAP-TLS authentication has the following guidelines and limitations:

- The IOS-XR software supports 802.1X only on physical ports (Ethernet interfaces).
- The IOS-XR software supports only EAP-TLS authentication method.
- 802.1X Port-based authentication is used only to derive keys for MKA, and does not perform port control.
- The IOS-XR software supports both the PAE roles, as an authenticator and a supplicant.
- The IOS-XR software as an authenticator supports Remote EAP authentication using RADIUS as EAP transport.

- The IOS-XR software supports only single-host mode, and not multi-host mode.

IEEE 802.1X Device Roles

The devices in the network have the following specific roles with IEEE 802.1X authentication.

- 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.
- Authenticator - An entity that facilitates authentication of other entities attached to the same LAN.
- Authentication Server - An entity that provides an authentication service to an Authenticator. This service determines, from the credentials provided by the Supplicant, whether the Supplicant is authorized to access the services provided by the system in which the Authenticator resides.

Prerequisites for MACSec MKA Using EAP-TLS Authentication

- Ensure that a Certificate Authority (CA) server is configured for the network.
- Ensure that the configured CA certificate is valid.
- Ensure that the user has configured Cisco Identity Services Engine (ISE) Release 2.2 onwards or Cisco Secure Access Control Server Release 5.6 onwards as external AAA server.
- Ensure that the remote AAA server is configured with EAP-TLS method.
- Ensure that both the routers, the CA server, and the external AAA server are synchronized using Network Time Protocol (NTP). If time is not synchronized on all these devices, certificates may not be validated.

MACsec with Local EAP-TLS 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 dot1x (802.1x) clients with the EAP-TLS method using TLS Version 1.0 (TLSv1). It provides EAP-TLS based mutual authentication, where a Master Session Key (MSK) is generated on successful authentication.

Configure MACSec Encryption Using EAP-TLS Authentication

Configuring MACSec encryption using EAP-TLS authentication involves the following tasks:

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. You can also specify an IPv6 address for the host (radius server).

```
Router# configure terminal
Router(config)# radius-server host 209.165.200.225 key 7 094F471A1A0A57
Router(config)# radius-server vsa attribute ignore unknown
Router(config)# commit
```

Running Configuration

```
Router# show run radius-server
radius-server host 209.165.200.225 auth-port 1646
  key 7 094F471A1A0A57
  radius-server vsa attribute ignore unknown
!
```

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.

```
Router# configure terminal
Router(config)# aaa authentication dot1x default group radius
Router(config)# commit
```

Running Configuration

```
Router# show run aaa
configure
  aaa authentication dot1x default group radius
```

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. You must enter the key modulus size when prompted.

```
Router# crypto key generate rsa 8002
Wed Aug  7 10:25:22.461 UTC
The name for the keys will be: 8002
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]: 600
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

To delete the RSA keys, use the `no` form: **no crypto key generate rsa**

The following is a sample output of **show crypto key mypubkey rsa** command.

```
Router# show crypto key mypubkey rsa

Key label: 8002
Type     : RSA General purpose
Size     : 600
Created  : 12:56:29 UTC Wed Aug 07 2019
Data     :
 3067300D 06092A86 4886F70D 01010105 00035600 3053024C 0096DB0F EE3B3233
```

```
 6E5FDA53 0FC504D1 9A056E29 BB703118 C6A8A254 1DC6504B 29CD4DA0 984735C8
 46CD39A1 C379B059 92870F76 693D4A66 D9953F69 450238D4 C57803AF 41160D4F
 C9451945 02030100 01
```

Running Configuration

You can also view the RSA keys in the running configuration. The keys in the following example are in OpenSSL format:

Note Only those keys that are generated in the `config` mode are visible in the running configuration.

```
Router(config)#crypto key generate rsa test
Router(config)#commit
Thu May 12 08:37:59.894 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nTbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOTycjsizI/TLBz9WoQX.rmxVwkNgTKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

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.

```
Router# configure terminal
Router(config)# crypto ca trustpoint test2
Router(config-trustp)# enrollment url http://caurl.com
Router(config-trustp)# subject-name CN=8000Series,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Router(config-trustp)# rsakeypair 8002
Router(config-trustp)# crl optional
Router(config-trustp)# commit
```

You can also specify the enrollment URL as an IP address (*http://10.2.2.2*).

Running Configuration

The following is a sample output of **show run crypto ca trustpoint test2** command.

```
crypto ca trustpoint test2
 crl optional
 subject-name CN=8000Series,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
 enrollment url http://caurl.com
 rsakeypair 8002
   !
```

Configure Domain Name

You can configure a domain name, which is required for certificate enrollment.

```
Router(config)# domain name ca.8000-series.cisco.com
Router(config)# commit
```

Running Configuration

The following is a sample output of **show run domain name** command.

```
Router# show run domain name
Thu Mar 29 16:10:42.533 IST
domain name ca.8000-series.cisco.com
```

Authenticate Certificate Authority and Request Certificates

Certificate enrollment 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.

```
Router# crypto ca authenticate test2
Router# crypto ca enroll test2
```

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 : test2
==================================================
CA certificate
Serial Number        : E0:18:F3:E4:53:17:3E:28
Subject              : subject-name CN=8002,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
```

```
Issued By            : subject-name CN=8002,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=209.165.200.225,unstructuredName=8002,
C=US,ST=NY,L=Newyork,O=Govt,OU=BU,CN=8002
Issued By            : CN=8000Series,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: test2
```

Configure EAP Profile

You can configure multiple EAP profiles.

```
Router# configure terminal
Router(config)# eap profile 8002
Router(config-eap)# identity CE1
Router(config-eap)# method tls pki-trustpoint test2
Router(config-eap)# commit
```

Running Configuration

The following is sample output of **show run eap** command.

```
Router# show run eap profile 8002
eap profile 8002
method tls pki-trustpoint test2
!
identity CE1
!
```

Configure 802.1X Profile on the Device

```
Router# configure
Router(config)# dot1x profile 8k_prof
Router(config-dot1x-8k_prof)# pae both
Router(config-dot1x-8k_prof)# authenticator timer reauth-time 3600
Router(config-dot1x-8k_prof)# supplicant eap profile 8002
Router(config-dot1x-8k_prof)# exit
Router(config)# commit
Router(config)# end
```

Running Configuration

The following is a sample output of the **show run dot1x profile 8k_prof** command.

```
Router# show run dot1x profile 8k_prof

dot1x profile 8k_prof
pae both
authenticator
  timer reauth-time 3600
!
```

```
supplicant
  eap profile 8002
!
```

Configure MACSec EAP and 802.1X Profile on an Interface

You can attach one of the 802.1X profiles on an interface.

```
Router# configure
Router(config)# interface fourHundredGigE 0/0/0/0
Router(config-if)# dot1x profile 8k_prof
Router(config-if)# macsec eap policy macsec-1
Router(config-if)# commit
```

Running Configuration

The following is a sample output of the **show run interface** command.

```
Router# show run interface HundredGigE 0/0/0/0
interface HundredGigE 0/0/0/0
dot1x profile 8k_prof
macsec eap policy macsec-1
!
```

Verify MACSec EAP and 802.1X Configuration on Interface

The following is a sample output of **show dot1x interface** command.

```
Router# show dot1x interface HundredGigE 0/0/0/24 detail

Dot1x info for HundredGigE 0/0/0/24
-----------------------------------------------------------------
Interface short name : Hu0/0/0/24
Interface handle     : 0x800020
Interface MAC        : 0201.9ab0.85af
Ethertype            : 888E
PAE                  : Both
Dot1x Port Status    : AUTHORIZED
Dot1x Profile        : 8k_prof
Supplicant:
Config Dependency    : Resolved
Eap profile          : 8k
Client List:
Authenticator        : 0257.3fae.5cda
EAP Method           : EAP-TLS
Supp SM State        : Authenticated
Supp Bend SM State   : Idle
Last authen time     : 2018 Mar 01 13:31:03.380
Authenticator:
Config Dependency    : Resolved
ReAuth               : Enabled, 0 day(s), 01:00:00
Client List:
Supplicant           : 0257.3fae.5cda
Auth SM State        : Authenticated
Auth Bend SM State   : Idle
Last authen time     : 2018 Mar 01 13:33:17.852
Time to next reauth  : 0 day(s), 00:59:57
MKA Interface:
Dot1x Tie Break Role : Auth
EAP Based Macsec     : Enabled
```

```
MKA Start time       : 2018 Mar 01 13:33:17.852
MKA Stop time        : NA
MKA Response time    : 2018 Mar 01 13:33:18.357
```

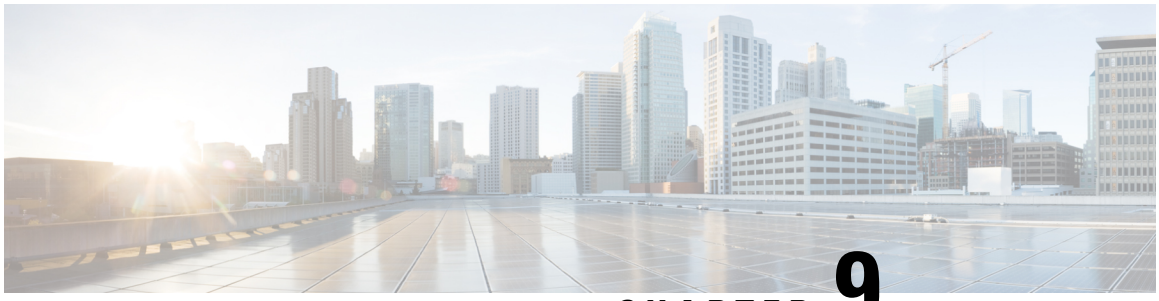
The following is a sample output of **show macsec mka session interface** command.

```
Router# show macsec mka session interface HundredGigE 0/0/0/24

============================================================================
Interface Local-TxSCI # Peers Status Key-Server
============================================================================
Hu0/0/0/24  0201.9ab0.85af/0001 1 Secured YES
```

The following is a sample output of **show macsec mka session interface detail** command.

```
Router# show macsec mka session interface HundredGigE 0/0/0/24 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                        : Hu0/0/0/24
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-XPN-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
```



CHAPTER 9

Implementing URPF

This section describes the implementation of URPF.

- Understanding uRPF, on page 229
- Configuring uRPF in Loose Mode, on page 230
- Configuring uRPF in Strict Mode, on page 232

Understanding uRPF

Table 34: Feature History Table

Feature Name	Release Information	
uRPF in Loose Mode	Release 7.3.15	When the source IP address of an incoming packet is not present in the Forwarding Information Base (FIB), the router considers it as an invalid packet and drops it. Use the **allow-default** keyword of **ipv4/ipv6 verify unicast source reachable-via** command and configure the default route for the interface so that the router does not drop a packet even when the source IP address is not present in the FIB. The command **ipv4/ipv6 verify unicast source reachable-via**is introduced.

It has become commonplace practice for hackers planning a Denial of Service (DoS) attack to use forged IP addresses (the practice is known as IP address spoofing). Hackers constantly change the source IP address to avoid detection by service providers. DoS uses more than one forged IP address from thousands of hosts that are infected with malware to flood a device. Therefore, it is complicated to identify and defeat the malware attack.

The uRPF is a mechanism for validating the source IP address of packets that are received on a router. A router that is configured with uRPF performs a reverse path lookup in the FIB table to validate the presence of the source IP address. If the FIB table lists the source IP address, then it indicates that the source is reachable and valid. If the FIB table does not list the source IP address, the router treats the packet as malicious and drops it.

The router supports uRPF in two modes:

- **uRPF in Loose Mode**: In uRPF loose mode, the router checks if it has a matching entry for the source IP address in the FIB and does not drop the legitimate regardless of interfaces the source address is learned on
- **uRPF in Strict Mode**: In uRPF strict mode, the router check if the interface receiving traffic packets is the same as the interface to reach the incoming packet's source address, the router considers such traffic packets legitimate and processes them. If not, the router drops it. The router supports uRPF in Strict Mode since IOS XR Release 7.9.1

Configuring uRPF in Loose Mode

When you configure uRPF in loose mode, the router checks if it has a matching entry for the source IP address in the FIB and does not drop the legitimate traffic that uses an alternate interface to reach the router. The uRPF in loose mode is useful in multihomed, service provider, edge networks.

Configuration

Use the following configuration to configure uRPF in loose mode on the router.

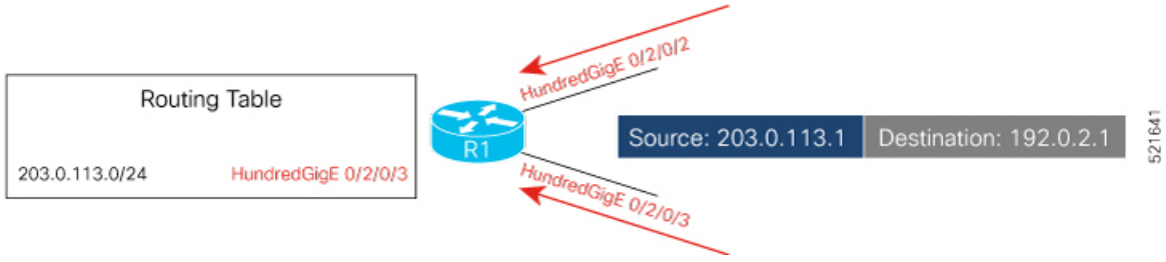
Note

- You can configure uRPF in loose mode on router interfaces, subinterfaces, bundle interfaces, and bundle subinterfaces
- Configure both IPv4 and IPv6 commands (as described in this section) for uRPF to work.

```
Router(config)# interface HundredGigE 0/2/0/2
Router(config-if)# ipv4 verify unicast source reachable-via any
Router(config-if)# ipv6 verify unicast source reachable-via any
Router(config-if)# commit
```

In the following figure, in router R1, the FIB table lists HundredGigE0/2/0/3 as the egress interface for the network 203.0.113.0/24. The ingress interface is HundredGigE 0/2/0/2. R1 receives packets with source IP address as 203.1.113.1 from both the interfaces, HundredGigE0/2/0/2 and HundredGigE0/2/0/3. When you configure uRPF in loose mode on the ingress interface, the router checks if the source address has a matching entry in the FIB table. The router does not drop the packet even if the ingress interface is not listed in the FIB tableas the outgoing interface for that prefix.

Figure 13: uRPF in Loose Mode



Running Configuration

To verify that the number of packets dropped due to uRPF configuration, you can use the **show cef drops**:

```
Router(config-if)# show cef drops
Node: 0/0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1911
  RPF suppressed drops packets :              0
  RP destined drops    packets :              0
  Discard drops        packets :              0
  GRE lookup drops     packets :              0
  GRE processing drops packets :              0
  LISP punt drops      packets :              0
  LISP encap err drops packets :              0
  LISP decap err drops packets :              0
Node: 0/RP0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1503
```

You have successfully configured uRPF in loose mode on the router.

Configuring Default Route for uRPF in Loose Mode

When you configure uRPF in loose mode, the source address of the packet must appear in the FIB for the verification process. However, you can use the **allow-default** option to use the default route in the source IP address verification process.

- When you do not configure the **allow-default** option, the router drops the packet that does not have its source address listed in the FIB table.
- When you configure the **allow-default** option, you must configure the default route for the interface. Otherwise, the router drops the packet.
- When you configure uRPF in loose mode with **allow-default** on any VRF interface, then it is applicable to all the interfaces in that VRF of the router.

Use the following configuration to configure uRPF in loose mode on the router along with the default address.

```
Router(config)# interface HundredGigE 0/2/0/2
Router(config-if)# ipv4 verify unicast source reachable-via any allow-default
Router(config-if)# ipv6 verify unicast source reachable-via any allow-default
Router(config-if)# commit
```

Configuring uRPF in Strict Mode

Table 35: Feature History Table

Feature Name	Release Information	Feature Description
uRPF in Strict Mode	Release 7.9.1	You can protect the router against DoS attacks with spoofed source IP addresses by enabling the Strict mode in uRPF. When this feature is enabled, the router accepts the incoming packet only if the source IP address of the packet is present in its routing table and if the source IP address of the input packet is reachable via the interface on which the packet has been received. If not, the router drops the packet.In earlier releases IOS XR supports only loose mode uRPF. This feature introduces the **hw-module profile cef unipath-surpf** command. This feature modifies the **ipv4/ipv6 verify unicast source reachable-via** command.

When you enable uRPF in strict mode, the router checks for the source address of the packet in the Forwarding Information Base (FIB). If the router receives the incoming packet on the same interface that the router would use to forward the traffic to the source of the packet, the packet passes the check and is further processed; otherwise, it is dropped. uRPF in strict mode should only be applied where there's natural or configured symmetry. Because internal interfaces are likely to have routing asymmetry. That is, multiple routes to the source of a packet, uRPF in strict mode shouldn't be implemented on interfaces that are internal to the network.

Usage Guidelines

- You can configure uRPF in strict mode on router interfaces, subinterfaces, bundle interfaces, and bundle subinterfaces.
- The tunnel and BVI interfaces don't support uRPF strict mode.
- Configure both IPv4 and IPv6 traffic types for uRPF to work.
- uRPF Strict mode is disabled in the router, by default.
- The uRPF in strict mode supports the **allow default** option. When the **allow default** option is enabled with the uRPFin strict mode, the packet is processed further only if it arrived through the default routes.

Prerequisites

Configure both IPv4 and IPv6 traffic types for uRPF to work.

Configuration

Use the following configuration to configure uRPF in strict mode on the router:

```
Router(config)# hw-module profile cef unipath-surpf enable
Router(config)# interface HundredGigE 0/2/0/2
Router(config-if)# ipv4 address 10.0.0.1 255.255.255.0
Router(config-if)# ipv4 verify unicast source reachable-via rx
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# ipv6 verify unicast source reachable-via rx
```

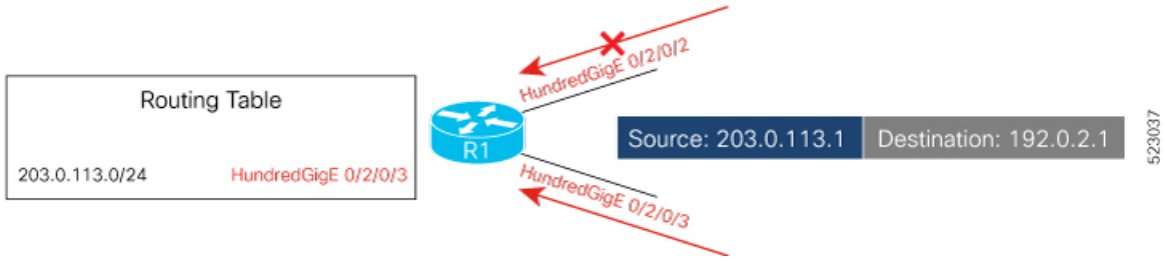
```
Router(config-if)# commit
Router(config-if)# exit
Router(config)# reload
```

Note You must reload the router after executing the **hw-module profile cef unipath-surpf** command.

In the following figure, in router R1, the FIB table lists HundredGigE0/2/0/3 as the egress interface for the network 203.0.113.0/24. R1 receives packets with source IP address as 203.1.113.1 from two different interfaces, HundredGigE0/2/0/2 and HundredGigE0/2/0/3. R1 accepts the packet coming from HundredGigE0/2/0/3 as the route to reach the source is 203.1.113.1 according to the FIB table. But the incoming packet via HundredGigE0/2/0/2 is dropped as the entries in the FIB table doesn't specifies HundredGigE0/2/0/2 as the interface to reach 203.1.113.1.

Note In the above example, the **hw-module profile cef unipath-surpf** configuration ensures the router R1 drops incoming packets via HundredGigE0/2/0/2, as according to the FIB table, the only interface to reach 203.0.113.0/24 is HundredGigE0/2/0/3. If there are multiple egress interfaces in router R1 for the 203.0.113.0/24 network, they will ensure to check all of these entries before dropping the packet.

Figure 14: uRPF in Strict Mode



Running Configuration

Confirm your configuration as shown:

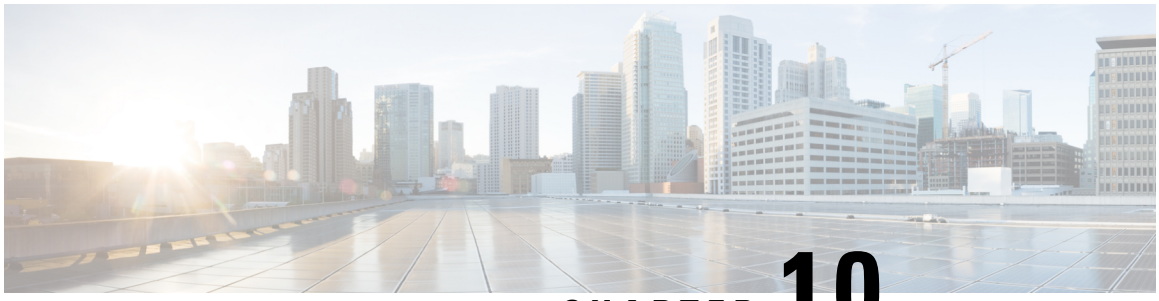
```
Router(config-if)# show running-config
...
!
interface HundredGigE 0/2/0/2
 ipv4 address 10.0.0.1 255.255.255.0
 ipv4 verify unicast source reachable-via rx
ipv6 address 2001::1/64
 ipv6 verify unicast source reachable-via rx
!
```

Running Configuration

To verify that the number of packets dropped due to uRPF configuration, you can use the **show cef drops**:

```
Router(config-if)# show cef drops
Node: 0/0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
```

```
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1911
  RPF suppressed drops packets :              0
  RP destined drops    packets :              0
  Discard drops        packets :              0
  GRE lookup drops     packets :              0
  GRE processing drops packets :              0
  LISP punt drops      packets :              0
  LISP encap err drops packets :              0
  LISP decap err drops packets :              0
Node: 0/RP0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1503
```



CHAPTER 10

Implementing Type 6 Password Encryption

Type 6 password encryption uses a reversible 128-bit AES encryption algorithm for storing passwords. Type 6 password encryption allows secure, and encrypted storage of plain-text passwords on the device. The device can decrypt the encrypted passwords into their original plain-text format.

You can use Type 6 password encryption to securely store plain text key strings for authenticating BGP, IP SLA, IS-IS, MACsec, OSPF, and RSVP sessions.

- How to Implement Type 6 Password Encryption , on page 235

How to Implement Type 6 Password Encryption

Scenario - The following 3-step process explains the Type 6 password encryption process for authenticating BGP sessions between two routers, R1 and R2.

Follow the first two steps for all Type 6 password encryption scenarios. The third step, *Creating BGP Sessions*, is specific to BGP. Similarly, you can enable Type 6 password encryption for OSPF, IS-IS, or other protocol sessions. For details on creating these protocol sessions, see the content in *Configure>Routing* listed here. For MACsec authentication, refer the Configuring MACsec, on page 159 chapter.

Note You must enable the master key for type 6 password encryption after an iPXE boot using the **key config-key password-encryption** command.

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

The Type6 encryption key, hereafter referred to as primary key in this chapter, is the password or key that encrypts all plain text key strings in the router configuration. An Advance Encryption Standard (AES) symmetric cipher does the encryption.

Creating the Primary Key

Use the **key config-key password-encryption** command to create the primary key.

Configuration Example

```
R1 & R2 # key config-key password-encryption
```

```
Fri Jul 19 12:22:45.519 UTC
New password Requirements: Min-length 6, Max-length 64
Characters restricted to [A-Z][a-z][0-9]
Enter new key :
Enter confirm key :
Master key operation is started in background
```

Once the command is executed, the **Master key operation**—creating, updating, or deleting the primary key—happens in the background. You can use the **show type6 server** command to view the status of the primary key operation.

When the key is created, it is stored internally; not as part of the router configuration. The router does not display the primary key as part of the running configuration. So, you cannot see or access the primary key when you connect to the router.

Enabling Type 6 Password Encryption

```
/* Enable Type 6 password encryption */
R1 & R2 (config)# password6 encryption aes
R1 & R2 (config)# commit
Fri Jul 19 12:22:45.519 UTC
```

Modifying the Primary Key

Note The Type 6 primary key update results in configuration change of the key chain and the other clients using Type 6. As the failure of router being configured can disrupt the product network, it is recommended to perform the primary key update operation during a maintenance window. Else, routing protocol sessions might fail.

The primary key is not saved to the running configuration, but the changes are persistent across reloads. The primary key update cannot be rolled back. That is, once the primary key is modified, you cannot revert to the older key using the **rollback configuration** command.

Enter the **key config-key password-encryption** command, and the old key and new key information.

```
R1 & R2# key config-key password-encryption

New password Requirements: Min-length 6, Max-length 64
Characters restricted to [A-Z][a-z][0-9]
Enter old key :
Enter new key :
Enter confirm key :
Master key operation is started in background
```

Deleting the Primary Key

```
R1 & R2# configure
R1 & R2 (config)# no password6 encryption aes
R1 & R2 (config)# commit
R1 & R2 (config)# exit
R1 & R2# key config-key password-encryption delete

WARNING: All type 6 encrypted keys will become unusable
Continue with master key deletion ? [yes/no]:yes
Master key operation is started in background
```

Verification

Verify that the primary key configuration and Type 6 feature configuration state are in the *Enabled* state. The **Master key Inprogress** field displays **No** to indicate that the primary key activity is complete (created, modified, or deleted). When you disable a primary key, **Disabled** is displayed for all the three states.

```
R1 & R2#show type6 server

Fri Jul 19 12:23:49.154 UTC
Server detail information:
==========================================
AES config State       :       Enabled
Masterkey config State :       Enabled
Type6 feature State    :       Enabled
Master key Inprogress  :       No
```

Verify Type 6 trace server details.

```
R1 & R2#show type6 trace server all

Fri Jul 19 12:26:05.111 UTC
Client file lib/type6/type6_server_wr
25 wrapping entries (18496 possible, 64 allocated, 0 filtered, 25 total)
Jul 19 09:59:27.168 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 ***** Type6 server process
started Respawn count (1) ****
…
…
Jul 19 12:22:59.908 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 User has started Master key
 operation (CREATE)
Jul 19 12:22:59.908 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Created Master key in TAM
successfully
Jul 19 12:23:00.265 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Master key Available set to
 (AVAILABLE)
Jul 19 12:23:00.272 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Master key inprogress set
to (NOT INPROGRESS)
```

From Cisco IOS XR Software Release 7.0.14 and later, you can use the **show type6 masterkey update status** command to display the update status of the primary key. Prior to this release, you could use the **show type6 clients** command for the same purpose.

```
Router#show type6 masterkey update status
Thu Sep 17 06:48:56.595 UTC
Type6 masterkey operation is NOT inprogress


Router#show type6 masterkey update status
Thu Sep 17 06:50:07.980 UTC
Type6 masterkey operation is inprogress

Masterkey upate status information:
Client Name               Status
====================================
keychain                  INPROGRESS
```

Clear Type 6 Client State

You can use the **clear type6 client** command in XR EXEC mode to clear the Type 6 client state.

If the primary key update operation is stuck at any stage, then you can use this **clear** command to clear that state. You can track the primary key update operation using the **show type6 server** command output. If the *Master key Inprogress* field in that output displays as *YES*, then you can use **show type6 masterkey update**

status command (or, **show type6 clients** command, prior to Release 7.0.14) 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 on key chains, refer the Implementing Keychain Management chapter.

If you enable Type 6 password encryption, plain-text keys are encrypted using Type 6 encryption. Enter plain-text key-string input in alphanumeric form. If you enable MACsec with Type 6 password encryption, the key-string input is in hexadecimal format.

Configuration

```
/* Enter the key chain details */
R1 & R2# configure
R1 & R2(config)# key chain my-test-keychain
R1 & R2(config-type6_password)# key 1
```

Enter the Type 6 encrypted format using the **key-string password6** command.

Note Using the **key-string** command, you can enter the password in clear text format or Type 6 encrypted (already encrypted password) format, as used in this scenario.

Note Enable the same key string for all the routers.

```
R1 & R2 (config-type6_password-1)# key-string password6
666449644369554484a4448674b695e685d56565d676364554b64555f4c5c645b
R1 & R2 (config-type6_password-1)# cryptographic-algorithm HMAC-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
```

Note Border Gateway Protocol (BGP) supports only HMAC-MD5 and HMAC-SHA1-12.

Verification

Verify key chain trace server information.

```
R1 & R2# show key chain trace server both

Sat Jul 20 16:44:08.768 UTC
```

```
Client file lib/kc/kc_srvr_wr
4 wrapping entries (18496 possible, 64 allocated, 0 filtered, 4 total)
Jul 20 16:43:26.342 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 *********kc_srvr process
started*********
Jul 20 16:43:26.342 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 (kc_srvr) Cerrno DLL registration
successfull
Jul 20 16:43:26.349 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 (kc_srvr) Initialised sysdb connection
Jul 20 16:43:26.612 lib/kc/kc_srvr_wr 0/RP0/CPU0 t317 (kc_srvr_type6_thread) Succesfully
registered as a type6 client
```

Verify configuration details for the key chain.

```
R1 & R2# show key chain type6_password

Sat Jul 20 17:05:12.803 UTC

Key-chain: my-test-keychain -
  Key 1 -- text "6664496443695544484a4448674b695e685d56565d676364554b64555f4c5c645b"
    Cryptographic-Algorithm -- HMAC_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 Cisco 8000 Series Routers*.

Configuration Example

```
/* Create BGP session on Router1 */
R1# configure
R1(config)# router bgp 65537
```

Ensure that you use the same key chain name for the BGP session and the Type 6 encryption (for example, *my-test-keychain* in this scenario).

```
R1 (config-bgp)# neighbor 10.1.1.11 remote-as 65537
R1 (config-bgp)# keychain my-test-keychain
R1 (config-bgp)# address-family ipv4 unicast
R1 (config-bgp)# commit
```

Repeat the above steps on Router 2 as well.

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

```
/* Create BGP session on Router2 */
R2 (config)# router bgp 65537
R2 (config-bgp)# neighbor 10.1.1.1 remote-as 65537
R2 (config-bgp)# keychain my-test-keychain
```

```
R2 (config-bgp)# address-family ipv4 unicast
R2 (config-bgp)# commit
```

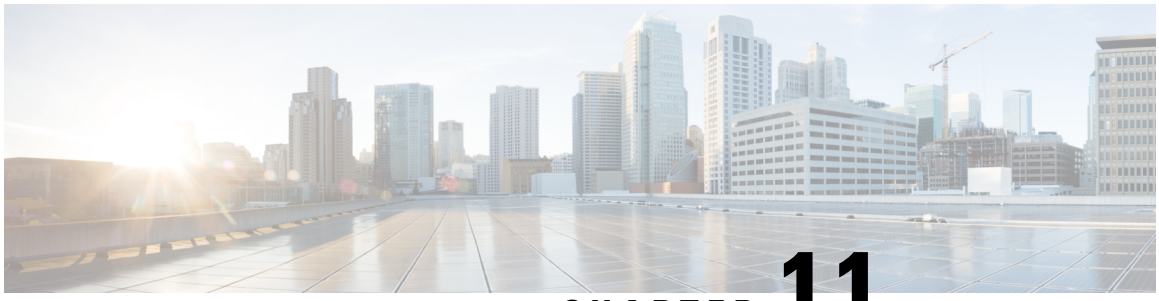
Verification

On the routers R1 and R2, verify that the BGP NBR state is in the *Established* state.

```
R1# show bgp sessions
Neighbor      VRF      Spk    AS      InQ  OutQ  NBRState     NSRState
10.1.1.11     default  0      65537   0    0     Established  None

R2# show bgp sessions
Neighbor      VRF      Spk    AS      InQ  OutQ  NBRState     NSRState
10.1.1.1     default  0      65537   0    0     Established  None
```

Associated Commands

- **session-group**
- **show bgp sessions**



CHAPTER 11

Implementing Management Plane Protection

The Management Plane Protection (MPP) feature in Cisco IOS XR software 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.

Device management traffic may enter a device only through these management interfaces. After MPP is enabled, no interfaces except designated management interfaces accept network management traffic destined to the device. Restricting management packets to designated interfaces provides greater control over management of a device, providing more security for that device.

This module describes how to implement management plane protection on Cisco 8000 Series Routers.

- Prerequisites for Implementing Management Plane Protection, on page 241
- Restrictions for Implementing Management Plane Protection, on page 241
- Information About Implementing Management Plane Protection, on page 242
- How to Configure a Device for Management Plane Protection, on page 244
- Configuration Examples for Implementing Management Plane Protection, on page 245
- Additional References, on page 247

Prerequisites for Implementing Management Plane Protection

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.

Restrictions for Implementing Management Plane Protection

- Currently, MPP does not keep track of 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.
- Route Processor (RP) interfaces are by default designated as out-of-band interfaces, and can be configured under MPP.

- 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), and HTTP.
- MPP does not support MIB.
- In an MPLS L3VPN, when MPP has a VRF interface attached, it applies the VRF filter on an incoming interface through LPTS. When an incoming packet from the core interface has a different VRF, then MPP does not allow it.

Note When configuring a device for MPP for an inband interface the **interface all** configuration does not apply specific VRF filter and allows traffic for all source and destination interfaces.

Information About Implementing Management Plane Protection

Before you enable the Management Plane Protection feature, you should understand the following concepts:

Inband Management Interface

An *inband management interface* is a Cisco IOS XR software 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*.

Out-of-Band Management 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.

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 Overview

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 IOS XR 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, HTTP, Secure HTTP (HTTPS), and SSH. 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.

Management Plane Protection Feature

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, using the MPP CLI that follows. Afterwards, only the default management interfaces and those you have previously configured as MPP interfaces will accept network management packets destined for the device. All other interfaces drop such packets.

Note Logical interfaces (or any other interfaces not present on the data plane) filter packets based on the ingress physical interface.

After configuration, you can modify or delete a management interface.

Following are the management protocols that the MPP feature supports. These management protocols are also the only protocols affected when MPP is enabled.

- SSHv2
- SNMP, all versions
- Telnet
- TFTP
- HTTP
- HTTPS

Benefits of the Management Plane Protection Feature

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.

How to Configure a Device for Management Plane Protection

This section contains the following tasks:

Configuring a Device for Management Plane Protection for an Inband 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.

```
Router#configure terminal
Router(config)#control-plane
Router(config-ctrl)#management-plane
Router(config-mpp)#inband
Router(config-mpp-inband)#interface fourHundredGigE 0/0/0/0
Router(config-mpp-inband-if)#allow telnet peer
Router(config-telnet-peer)#address ipv4 10.1.0.0/16
Router(config-telnet-peer)#commit
```

- FourHundredGigE 0/0/0/0 is configured as an inband interface. Use the **interface all** command form to configure all interfaces as inband interfaces.
- Telnet protocol is configured on the inband interface. To enable all protocols, use the **allow all** command form.

Running Configuration

The following is a sample output of **show mgmt-plane** command for the inband interface fourHundredGigE 0/0/0/0.

```
Router#show mgmt-plane inband interface fourHundredGigE 0/0/0/0

interface - fourHundredGigE 0/0/0/0
        telnet configured -
                peer v4 allowed - 10.1.0.0/16
```

Configuring a Device for Management Plane Protection for an Out-of-band Interface

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.
- For a specific protocol, configure the protocol VRF server for the out-of-band VRF.

```
Router#configure terminal
Router(config)#control-plane
Router(config-ctrl)#management-plane
Router(config-mpp)#out-of-band
Router(config-mpp-outband)#vrf target
Router(config-mpp-outband)#interface fourHundredGigE 0/0/0/3
Router(config-mpp-outband-if)#allow tftp peer
Router(config-tftp-peer)#address ipv6 33::33
Router(config-tftp-peer)#commit
```

- FourHundredGigE 0/0/0/3 is configured as an out-of-band interface, for the VRF **target**. Use the **interface all** command form to configure all interfaces as out-of-band interfaces.
- TFTP protocol is configured on the out-of-band interface. To enable all protocols, use the **allow all** command form.

Running Configuration

The following is a sample output of the **show mgmt-plane out-of-band vrf** command.

```
Router#show mgmt-plane out-of-band vrf

Management Plane Protection -
        out-of-band VRF - target
```

Configuration Examples for Implementing Management Plane Protection

This section provides the following configuration example:

Configuring Management Plane Protection: Example

The following example shows a detailed example of how to configure inband and out-of-band interfaces under MPP:

```
configure
 control-plane
  management-plane
   inband
    interface all
     allow SSH
     !
    interface fourHundredGigE 0/0/0/0
     allow all
     allow SSH
     allow Telnet peer
      address ipv4 10.1.0.0/16
    !
   !
    interface fourHundredGigE 0/0/0/0
     allow Telnet peer
      address ipv4 10.1.0.0/16
     !
    !
   !
   out-of-band
    vrf target
    interface fourHundredGigE 0/0/0/3
     allow TFTP peer
      address ipv6 33::33
     !
    !
   !
  !
 !

show mgmt-plane

Management Plane Protection

inband interfaces
----------------------

interface - fourHundredGigE 0/0/0/0
        ssh configured -
                All peers allowed
        telnet configured -
                peer v4 allowed - 10.1.0.0/16
        all configured -
                All peers allowed
interface - fourHundredGigE 0/0/0/0
        telnet configured -
                peer v4 allowed - 10.1.0.0/16

interface - all
        all configured -
                All peers allowed

outband interfaces
----------------------
interface - fourHundredGigE 0/0/0/3
```

```
        tftp configured -
                peer v6 allowed - 33::33

show mgmt-plane out-of-band vrf

Management Plane Protection -
        out-of-band VRF - target
```

Additional References

The following sections provide references related to implementing management plane protection.

Related Documents

Related Topic	Document Title
MPP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Management Plane Protection Commands* on *System Security Command Reference for Cisco 8000 Series Routers*.

Standards

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

MIBs

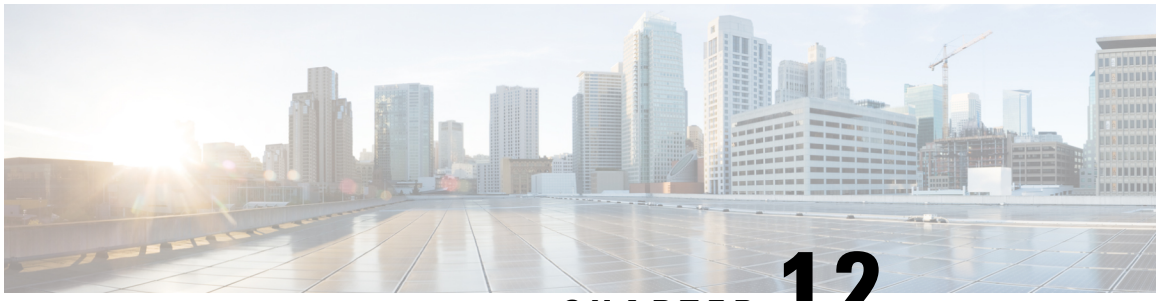
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature.	—

Technical Assistance

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



CHAPTER 12

Implementing Secure Shell

Secure Shell (SSH) is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley **rexec** and **rsh** tools.

Two versions of the SSH server are available: SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2). SSHv1 uses Rivest, Shamir, and Adelman (RSA) keys and SSHv2 uses either Digital Signature Algorithm (DSA) keys or RSA keys, or Elliptic Curve Digital Signature Algorithm (ECDSA) keys. Cisco IOS XR software supports both SSHv1 and SSHv2.

This module describes how to implement Secure Shell on Cisco 8000 Series Routers.

Note Cisco IOS XR does not support X11 forwarding through an SSH connection.

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.

Note For a complete description of the Secure Shell commands used in this chapter, see the *Secure Shell and Secure Socket Layer Commands* chapter in *System Security Command Reference for Cisco 8000 Series Routers*.

Release	Modification
Release 7.3.2	Introduced CiscoSSH.
Release 7.3.2	Introduced SSH port forwarding feature with CiscoSSH.
Release 7.3.15	Introduced SSH port forwarding feature with Cisco IOS XR SSH.

Release	Modification
Release 7.3.1	Introduced these features: • Ed25519 Public-Key Algorithm Support for SSH • User Configurable Maximum Authentication Attempts for SSH • X.509v3 Certificate-based Authentication for SSH
Release 7.0.12	This chapter was introduced.

• Information About Implementing Secure Shell, on page 250
• Prerequisites for Implementing Secure Shell, on page 260
• Guidelines and Restrictions for Implementing Secure Shell, on page 261
• How to Implement Secure Shell, on page 262

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 IOS XR software authentication. The SSH server in Cisco IOS XR 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 in the Cisco IOS XR software works with publicly and commercially available SSH servers. The SSH client supports the ciphers of AES, 3DES, the hash algorithm SHA1, and password authentication. 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 using this command:

ssh client dscp *dscp-value*

The *dscp-value* ranges from 0 to 63. If not configured, 16 is set as the default DSCP value in the packets (for both client and server).

You can use the **ssh client** command in the XR Config mode to configure various SSH client options.

SSH also supports remote command execution as follows:

```
Router#ssh 192.0.2.1 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)
Router#
```

SFTP Feature Overview

SSH includes support for secure 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 The default policer value for SSH-Known is set to 300pps. Slower transfers are expected due to this. You can adjust the lpts policer value for this punt cause to higher values that allows faster transfers.

You can increase the throughput of SCP or SFTP over inband using the **ssh server tcp-window-scale** command.

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 SFTP server instances are created. The

SSH 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 key pair on the client machine. The private key of the RSA key pair remains on the client machine.

The user generates an RSA public-private key pair on a unix client using a standard key generation mechanism such as ssh-keygen. The max length of the keys supported is 4096 bits, and the minimum length is 512 bits. The following example displays a typical key generation activity:

```
bash-2.05b$ ssh-keygen -b 1024 -t rsa
Generating RSA private key, 1024 bit long modulus
```

The public key must be in base64 encoded (binary) format for it to be imported correctly into the box. 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 supports the RSA based authentication. For more information on how to import the public key to the router, see the *Implementing Certification Authority Interoperability* chapter in this guide.

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 provides the primary framework through which access control can be set up on your Cisco router or access server. For more information on AAA, the *Configuring AAA Services* chapter in this guide.

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.

SSH and SFTP in Baseline Cisco IOS XR Software Image

The SSH and SFTP components are present in the baseline Cisco IOS XR software image. The management and control plane components (such as the IPSec control plane) are also present in the base package. However, the data plane components (such as the MACSec and the IPSec data plane) are 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.

CiscoSSH

Table 36: Feature History Table

Feature Name	Release Information	Feature Description
CiscoSSH	Release 7.3.2	This release introduces CiscoSSH, a newer implementation of SSH on this platform. 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. **Note** Cisco IOS XR SSH, the SSH implementation that existed prior to this release, is now deprecated.

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 details on SSH in general, see Information About Implementing Secure Shell, on page 250 and How to Implement Secure Shell, on page 262.

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 256.

Table 37: 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 ':' (*colon*) as a delimiter in certain types of user authentication, it does not support authentication of usernames having ':' (*colon*) in it.	No restriction for using ':' (*colon*) 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.
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.

Functionality	CiscoSSH	Cisco IOS XR SSH
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 support management access to the router over the standby management Ethernet interface.
- 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**
- If you configure ingress ACLs only under the management interface and do not configure them under the **ssh server** configuration mode, then those ingress ACLs do not have any impact on the SSH, or Netconf traffic. This behavior is applicable only to ingress ACLs attached to management interface.

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 not 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.
- **Ingress ACL:** To filter out the ingress SSH and Netconf traffic, we recommend to configure the ingress ACL under the **ssh server** configuration mode instead of configuring under the management interface.

  For SSH:

  ssh server vrf *vrf-name* ipv4 access-list *ipv4-access-list-name* ipv6 access-list *ipv6-access-list-name*

  For Netconf:

  ssh server netconf vrf *vrf-name* ipv4 access-list *ipv4-access-list-name* ipv6 access-list *ipv6-access-list-name*
- **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 38: 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 l1/l2/l3** command. Similarly, you can use the **debug ssh client l1/l2/l3** 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 39: Syslogs for CiscoSSH and Cisco IOS XR SSH

Session Event	Syslogs on CiscoSSH	Syslogs on Cisco IOS XR SSH
Session login	RP/0/RP0/CPU0:Sep 22 11:06:33.467 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PRX-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_PRX-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_PRX-6-INFO_GENERAL : sshd[32564]: Starting session: shell on pts/1 for admin from 203.0.113.1 port 62015 id 0	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')

Session Event	Syslogs on CiscoSSH	Syslogs on Cisco IOS XR SSH
Session logout	RP/0/RP0/CPU0:Sep 22 11:11:27.394 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PRX-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_PRX-6-INFO_GENERAL : sshd[32564]: Disconnected from user admin 203.0.113.1 port 62015	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'
Session login failure	RP/0/RP0/CPU0:Sep 22 19:47:06.211 IST: ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[31103]: Failed authentication/pam for admin from 203.0.113.1 port 60189 ssh2	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'
Session rekey	ssh_syslog_proxy[1204]: %SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[24919]: Server initiated time rekey for session=21, session_rekey_count =1	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

Prerequisites for Implementing Secure Shell

The following prerequisites are required to implement Secure Shell:

- 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.
- To run an SSHv2 server, you must have a VRF. This may be the default VRF or a specific VRF. VRF changes are applicable only to the SSH v2 server.
- Configure user authentication for local or remote access. You can configure authentication with or without authentication, authorization, and accounting (AAA). For more information, see the *Configuring AAA Services* chapter in the this guide.
- AAA authentication and authorization must be configured correctly for Secure Shell File Transfer Protocol (SFTP) to work.

Guidelines and Restrictions for Implementing Secure Shell

The following are some basic SSH guidelines, restrictions, and limitations of the SFTP feature:

- In order for an outside client to connect to the router, the router needs to have an RSA (for 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.

Note The RSA, DSA and ECDSA keys are auto-generated during the boot if there is no key present.

- If you delete all the default crypto keys (the keys with **the_default** label) on the router, the SSH clients cannot establish sessions with the router. Hence, for clients to successfully establish SSH sessions with the router, ensure that at least one default crypto key is always present on the router. In FIPS mode, it is mandatory to have at least one default crypto key of type RSA or ECDSA.
- For SSH sessions, the router supports key-exchange algorithms (**diffie-hellman-group1-sha1** and **curve25519**) and cipher algorithms (**3des-cbc** and **chacha20-poly1305@openssh.com**) only in non-FIPS mode. The SSH session fails to connect if any of these algorithms is pre-configured prior to enabling FIPS mode though.
- 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, SFTP and SCP servers. The support however, is not extended to the SSH client.
- Execution shell, SFTP, SCP and Netconf are the only applications supported.
- The cipher preference for the SSH server follows the order AES128, AES192, AES256, aes128-gcm, aes256-gcm, and chacha20-poly1305. 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 unsupported, and the software generates a warning message in this case.
- Password messages of “none” are unsupported on the SSH client.
- Because the router infrastructure does not provide support for UNIX-like file permissions, files created on the local device lose the original permission information. 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.

How to Implement Secure Shell

To configure SSH, perform the tasks described in the following sections:

Configure SSH

Note For SSHv1 configuration, Step 1 to Step 4 are required. For SSHv2 configuration, these steps are optional.

Perform this task to configure SSH.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **hostname** *hostname*

Example:

```
Router(config)# hostname router1
```

Configures a hostname for your router.

Step 3 **domain name** *domain-name*

Example:

```
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 **configure**

Step 6 **ssh server tcp-window-scale** *scale*

Example:

```
Router(config)# ssh server tcp-window-scale 10
```

(Optional) Configures the TCP window scale for increased throughput for SCP or SFTP.

Step 7 **ssh timeout** *seconds*

Example:

```
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 8 Do one of the following:

- **ssh server** [**vrf** *vrf-name* [**ipv4 access-list** *ipv4-access-list name*] [**ipv6 access-list** *ipv6-access-list name*]]
- **ssh server v2**

Example:

```
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. Optionally ACLs for IPv4 and IPv6 can be used to restrict access to the server before the port is opened. 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 9 **ssh** {**client** | **server**} **dscp** *dscp-value*

Example:

```
Router(config)# ssh server dscp 63
```

```
Router(config)# ssh client dscp 63
```

(optional) Sets the DSCP value in the outgoing packets. If not configured, 16 is set as the default DSCP value for the packets (for both client and server).

Step 10 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 11 **show ssh**

Example:

```
Router# show ssh
```

(Optional) Displays all of the incoming and outgoing SSHv1 and SSHv2 connections to the router.

Step 12 **show ssh session details**

Example:

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

(Optional) Displays a detailed report of the SSHv2 connections to and from the router.

Step 13 **show ssh history**

Example:

```
Router# show ssh history
```

(Optional) Displays the last hundred SSH connections that were terminated.

Step 14 **show ssh history details**

Example:

```
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 15 **show tech-support ssh**

Example:

```
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.

Although the host keys 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 **crypto key zeroize** command in XR EXEC mode to remove the algorithms that are not required.

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.

If the SSH host-key pairs are not present in some scenarios, you can execute the **crypto key generate** command in XR EXEC mode to generate the required host-key pairs.

Configure the Allowed SSH Host-Key Pair Algorithms

When the SSH client attempts a connection with the SSH server, it sends a list of SSH host-key pair algorithms (in the order of preference) internally in the connection request. The SSH server, in turn, picks the first matching algorithm from this request list. The server establishes a connection only if that host-key pair is already generated in the system, and if it is configured (using the **ssh server algorithms host-key** command) as the allowed algorithm.

Note If this configuration of allowed host-key pairs is not present in the SSH server, then you can consider that the SSH server allows all host-key pairs. In that case, the SSH client can connect with any one of the host-key pairs. Not having this configuration also ensures backward compatibility in system upgrade scenarios.

Configuration Example

You may perform this (optional) task to specify the allowed SSH host-key pair algorithm (in this example, **ecdsa**) from the list of auto-generated host-key pairs on the SSH server:

```
/* Example to select the ecdsa algorithm */
Router(config)#ssh server algorithms host-key ecdsa-nistp521
```

Similarly, you may configure other algorithms.

Running Configuration

```
ssh server algorithms host-key ecdsa-nistp521
!
```

Verify the SSH Host-Key Pair Algorithms

Note With the introduction of the automatic generation of SSH host-key pairs, the output of the **show crypto key mypubkey** command displays key information of all the keys that are auto-generated. Before its introduction, the output of this show command displayed key information of only those keys that you explicitly generated using the **crypto key generate** command.

```
Router#show crypto key mypubkey ecdsa
Mon Nov 19 12:22:51.762 UTC
Key label: the_default
Type     : ECDSA General Curve Nistp256
Degree   : 256
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04AC7533 3ABE7874 43F024C1 9C24CC66 490E83BE 76CEF4E2 51BBEF11 170CDB26
14289D03 6625FC4F 3E7F8F45 0DA730C3 31E960FE CF511A05 2B0AA63E 9C022482
6E

Key label: the_default
Type     : ECDSA General Curve Nistp384
Degree   : 384
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04B70BAF C096E2CA D848EE72 6562F3CC 9F12FA40 BE09BFE6 AF0CA179 F29F6407
FEE24A43 84C5A5DE D7912208 CB67EE41 58CB9640 05E9421F 2DCDC41C EED31288
6CACC8DD 861DC887 98E535C4 893CB19F 5ED3F6BC 2C90C39B 10EAED57 87E96F78
B6

Key label: the_default
Type     : ECDSA General Curve Nistp521
Degree   : 521
Created  : 10:59:09 UTC Mon Nov 19 2018
Data     :
0400BA39 E3B35E13 810D8AE5 260B8047 84E8087B 5137319A C2865629 8455928F
D3D9CE39 00E097FF 6CA369C3 EE63BA57 A4C49C02 B408F682 C2153B7F AAE53EF8
A2926001 EF113896 5F1DA056 2D62F292 B860FDFB 0314CE72 F87AA2C9 D5DD29F4
DA85AE4D 1CA453AC 412E911A 419E9B43 0A13DAD3 7B7E88E4 7D96794B 369D6247
E3DA7B8A 5E
```

The following example shows the output for **ed25519**:

```
Router#show crypto key mypubkey ed25519
Wed Dec 16 16:12:21.464 IST
Key label: the_default
Type     : ED25519
Size     : 256
Created  : 15:08:28 IST Tue Oct 13 2020
Data     :
 649CC355 40F85479 AE9BE26F B5B59153 78D171B6 F40AA53D B2E48382 BA30E5A9

Router#
```

Related Topics

Automatic Generation of SSH Host-Key Pairs, on page 265

Associated Commands

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

Ed25519 Public-Key Signature Algorithm Support for SSH

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
Ed25519 Public-Key Signature Algorithm Support for SSH	Release 7.3.1	This algorithm is now supported on Cisco IOS XR 64-bit platforms when establishing SSH sessions. It is a modern and secure public-key signature algorithm that provides several benefits, particularly resistance against several side-channel attacks. Prior to this release, DSA, ECDSA, and RSA public-key algorithms were supported. This command is modified for this feature: ssh server algorithms host-key

This feature introduces the support for Ed25519 public-key algorithm, when establishing SSH sessions, on Cisco IOS XR 64-bit platforms. This algorithm offers better security with faster performance when compared to DSA or ECDSA signature algorithms.

The order of priority of public-key algorithms during SSH negotiation between the client and the server is:

- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384

- ecdsa-sha2-nistp521
- ssh-ed25519
- ssh-rsa
- ssh-dsa

Restrictions for ED25519 Public Key for SSH

The Ed25519 public key algorithm is not FIPS-certified. That is, if FIPS mode is enabled on the router, the list of public-key algorithms sent during the SSH key negotiation phase does not contain the Ed25519 key. This behavior is applicable only for new SSH connections. Any existing SSH session that has already negotiated Ed25519 public-key algorithm remains intact and continues to execute until the session is disconnected.

Further, if you have configured the router to negotiate only the Ed25519 public-key algorithm (using the **ssh server algorithms host-key** command), and if FIPS mode is also enabled, then the SSH connection to the router fails.

How to Generate Ed25519 Public Key for SSH

To generate Ed25519 public key for SSH, see Generate Crypto Key for Ed25519 Signature Algorithm, on page 142.

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 Configure the Allowed SSH Host-Key Pair Algorithms, on page 265.

To remove the Ed25519 key from the router, use the **crypto key zeroize ed25519** command in XR EXEC mode.

Configure the 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:

```
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* | *hostname*} [**username** *user- id* | **cipher des** | **source-interface** *type instance*]

Example:

```
Router# ssh remotehost username user1234
```

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 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 or RSA or ECDSA key pair might have been zeroed out. Make sure you follow similar steps as mentioned above to generate the required 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 in that case.

  - No domain specified

  You must configure a host domain for the router using the **domain-name** command in that case.

- 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. The default number of VTYs is 5. So, you must configure the number of VTYs in the VTY pool. The default value for the maximum number of SSH sessions is 64.
- For FIPS compliance,the weaker ciphers like 3DES and AES CBC are not supported; only AES-CTR cipher is supported.
- SSH uses either local authentication or remote authentication 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.

Order of SSH Client Authentication Methods

The default order of authentication methods for SSH clients on Cisco IOS XR routers is as follows:

- On routers running Cisco IOS XR SSH:
  - **public-key**, **password** and **keyboard-interactive**
- On routers running CiscoSSH (open source-based SSH):
  - **public-key**, **keyboard-interactive** and **password**

How to Set the Order of Authentication Methods for SSH Clients

To set the preferred order of authentication methods for SSH clients on Cisco IOS XR routers, use the **ssh client auth-method** command in the XR Config mode. This command is available from Cisco IOS XR Software Release 7.9.2/Release 7.10.1and later.

Configuration Example

In this example, we set the order of SSH client authentication methods in such a way that public key authentication is negotiated first, followed by keyboard-interactive, and then password-based authentication.

```
Router#configure
Router(config)#ssh client auth-method public-key keyboard-interactive password
Router(config-ssh)#commit
```

Running Configuration

```
Router#show run ssh client auth-methods
Tue Nov 21 17:55:44.688 IST
ssh client auth-methods public-key keyboard-interactive password
Router#
```

Configure Secure Shell: Example

This 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.

```
configure
hostname router1
domain name cisco.com
exit
configure
ssh server
end
```

Multi-channeling in SSH

The multi-channeling (also called multiplexing) feature on the Cisco IOS XR software server allows you to establish multiple channels over the same TCP connection from the SSH clients originating from the same host. Thus, rather than opening a new TCP socket for each SSH connection, all the SSH connections are multiplexed into one TCP connection and a single SSH session. For example, with multiplexing support on your XR software server, on a single SSH connection you can simultaneously open a pseudo terminal, remotely execute a command and transfer a file using any file transfer protocol. Multiplexing offers the following benefits:

- You are required to authenticate only once at the time of creating the session. After that, all the SSH clients associated with a particular session use the same TCP socket to communicate to the server.
- Saves time consumed otherwise wasted in creating a new connection each time.

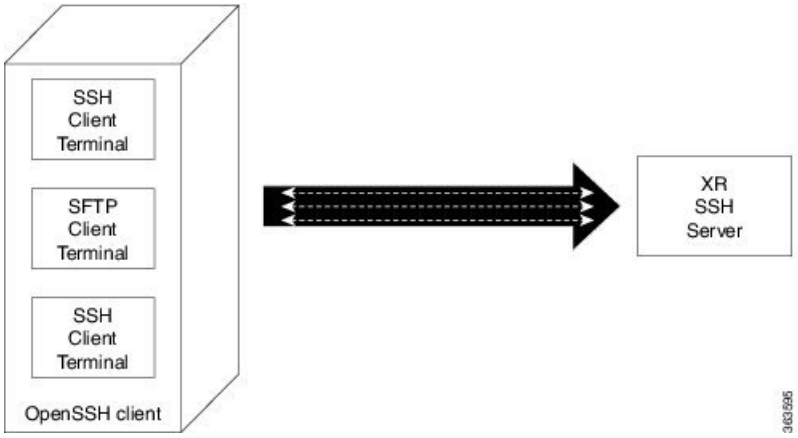
Multiplexing is enabled by default in the Cisco IOS XR software server. If your client supports multiplexing, you must explicitly set up multiplexing on the client for it to be able to send multi-channel requests to the server. You can use OpenSSH, Perl, WinSCP, FileZilla, TTSSH, Cygwin or any other SSH-based tool to set up multiplexing on the client. See Configure Client for Multiplexing, on page 272 provides an example of how you can configure the client for multiplexing using OpenSSH.

Restrictions for Multi-channeling Over SSH

- Do not use client multiplexing for heavy transfer of data as the data transfer speed is limited by the TCP speed limit. Hence, for a heavy data transfer it is advised that you run multiple SSH sessions, as the TCP speed limit is per connection.
- Client multiplexing must not be used for more than 15 concurrent channels per session simultaneously.

Client and Server Interaction Over Multichannel Connection

The figure below provides an illustration of a client-server interaction over a SSH multichannel connection.



As depicted in the illustration,

- The client multiplexes the collection of channels into a single connection. This allows different operations to be performed on different channels simultaneously. The dotted lines indicate the different channels that are open for a single session.
- After receiving a request from the client to open up a channel, the server processes the request. Each request to open up a channel represents the processing of a single service.

Note The Cisco IOX software supports server-side multiplexing only.

Configure Client for Multiplexing

The SSH client opens up one TCP socket for all the connections. In order to do so, the client multiplexes all the connections into one TCP connection. Authentication happens only once at the time of creating the session. After that, all the SSH clients associated with the particular session uses the same TCP socket to communicate to the server. Use the following steps to configure client multiplexing using OpenSSH:

Procedure

Step 1 Edit the ssh_config file.

Open the ssh_config file with your favorite text editor to configure values for session multiplexing. The system-wide SSH configuration file is located under /etc/ssh/ssh_config. The user configuration file is located under ~/.ssh/config or $HOME/.ssh/config.

Step 2 Add entries **ControlMaster auto** and **ControlPath**

Add the entry ControlMaster auto and ControlPath to the ssh_config file, save it and exit.

- ControlMaster determines whether SSH will listen for control connections and what to do about them. Setting the ControlMaster to 'auto' creates a primary session automatically but if there is a primary session already available, subsequent sessions are automatically multiplexed.
- ControlPath is the location for the control socket used by the multiplexed sessions. Specifying the ControlPath ensures that any time a connection to a particular server uses the same specified primary connection.

Example:

```
Host *
ControlMaster auto
ControlPath ~/.ssh/tmp/%r@%h:%p
```

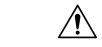
Step 3 Create a temporary folder.

Create a temporary directory inside the /.ssh folder for storing the control sockets.

SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm

The Cisco IOS XR software provides a new configuration option to control the key algorithms to be negotiated with the peer while establishing an SSH connection with the router. With this feature, you can enable the insecure SSH algorithms on the SSH server, which are otherwise disabled by default. A new configuration option is also available to restrict the SSH client from choosing the HMAC, or hash-based message authentication codes algorithm while trying to connect to the SSH server on the router.

You can also configure a list of ciphers as the default cipher list, thereby having the flexibility to enable or disable any particular cipher.



Caution Use caution in enabling the insecure SSH algorithms to avoid any possible security attack.

To disable the HMAC algorithm, use the **ssh client disable hmac** command or **ssh server disable hmac** command in XR Config mode.

To enable the required cipher, use the **ssh client enable cipher** command or the **ssh server enable cipher** command in XR Config mode.

The supported encryption algorithms (in the order of preference) are:

1. aes128-ctr
2. aes192-ctr
3. aes256-ctr
4. aes128-gcm@openssh.com
5. aes256-gcm@openssh.com
6. aes128-cbc
7. aes192-cbc
8. aes256-cbc
9. 3des-cbc

In SSH, the CBC-based ciphers are disabled by default. To enable these, you can use the **ssh client enable cipher** command or the **ssh server enable cipher** command with the respective CBC options (aes-cbc or 3des-cbc). All CTR-based and GCM-based ciphers are enabled by default.

Disable HMAC Algorithm

Configuration Example to Disable HMAC Algorithm

```
Router(config)# ssh server disable hmac hmac-sha1
Router(config)#commit
```

```
Router(config)# ssh client disable hmac hmac-sha1
Router(config)#commit
```

Running Configuration

```
ssh server disable hmac hmac-sha1
!
```

```
ssh client disable hmac hmac-sha1
!
```

Related Topics

SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 273

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 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 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 273

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 41: Feature History Table

Feature Name	Release Information	Feature Description
User Configurable Maximum Authentication Attempts for SSH	Release 7.3.1	This feature allows you to set a limit on the number of user authentication attempts allowed for SSH connection, using the three authentication methods that are supported by Cisco IOS XR. The limit that you set is an overall limit that covers all the authentication methods together. If the user fails to enter the correct login credentials within the configured number of attempts, the connection is denied and the session is terminated. This command is introduced for this feature: ssh server max-auth-limit

The three SSH authentication methods that are supported by Cisco IOS XR are public-key (which includes certificate-based authentication), keyboard-interactive, and password authentication. The limit count that you set as part of this feature comes into effect whichever combination of authentication methods you use. The limit ranges from 3 to 20; default being 20 (prior to Cisco IOS XR Software Release 7.3.2, the limit range was from 4 to 20).

Restrictions for Configuring Maximum Authentication Attempts for SSH

These restrictions apply to configuring maximum authentication attempts for SSH:

- This feature is available only for Cisco IOS XR routers functioning as SSH server; not for the ones functioning as SSH clients.
- This configuration is not user-specific; the limit remains same for all the users.

- Due to security reasons, the SSH server limits the number of authentication attempts that explicitly uses the password authentication method to a maximum of 3. You cannot change this particular limit of 3 by configuring the maximum authentication attempts limit for SSH.

  For example, even if you configure the maximum authentication attempts limit as 5, the number of authentication attempts allowed using the password authentication method still remain as 3.

Configure Maximum Authentication Attempts for SSH

You can use the **ssh server max-auth-limit** command to specify the maximum number of authentication attempts allowed for SSH connection.

Configuration Example

```
Router#configure
Router(config)#ssh server max-auth-limit 5
Router(config)#commit
```

Running Configuration

```
Router#show running-configuration ssh
ssh server max-auth-limit 5
ssh server v2
!
```

Verification

The system displays the following SYSLOG on the router console when maximum authentication attempts is reached:

```
RP/0/RP0/CPU0:Oct 6 10:03:58.029 UTC: SSHD_[68125]: %SECURITY-SSHD-3-ERR_GENERAL : Max
authentication tries reached-exiting
```

Associated Commands

- **ssh server max-auth-limit**

X.509v3 Certificate-based Authentication for SSH

Table 42: Feature History Table

Feature Name	Release Information	Feature Description
X.509v3 Certificate-based Authentication for SSH	Release 7.3.1	This feature adds new public-key algorithms that use X.509v3 digital certificates for SSH authentication. These certificates use a chain of signatures by a trusted certification authority to bind a public key to the digital identity of the user who is authenticating with the SSH server. These certificates are tough to falsify and are therefore used for identity management and access control across many applications and networks. Commands introduced for this feature are: ssh server certificate ssh server trustpoint This command is modified for this feature: ssh server algorithms host-key

This feature support is available for the SSH server for the server authentication and the 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.

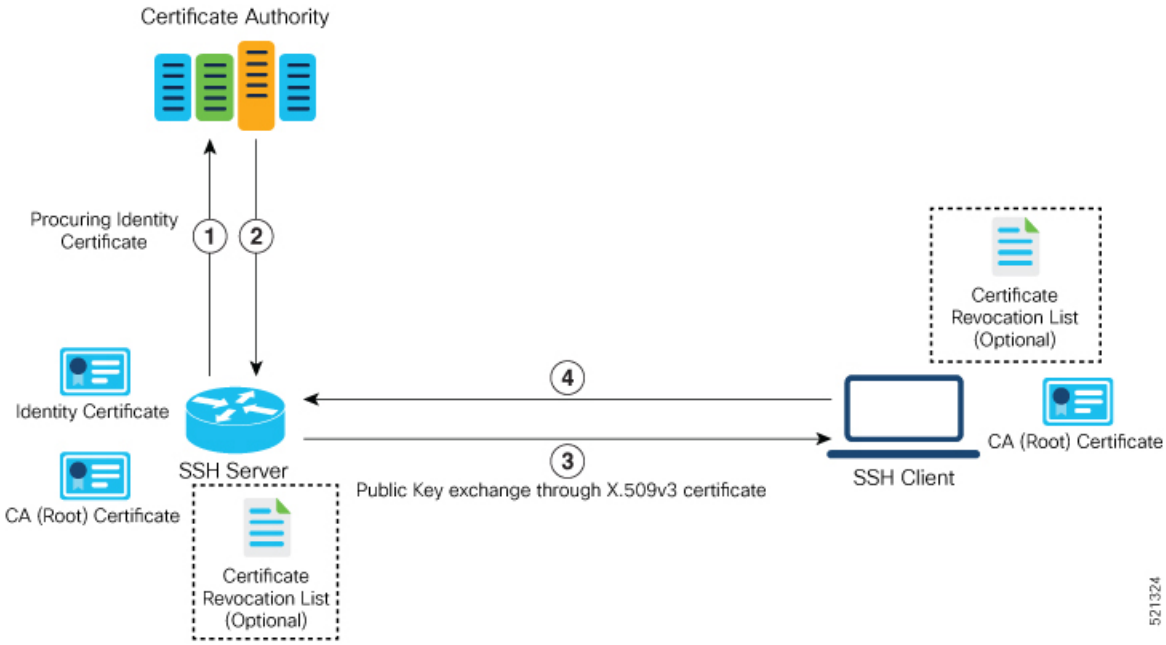
There are two SSH protocols that use public-key cryptography for authentication:

- 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.

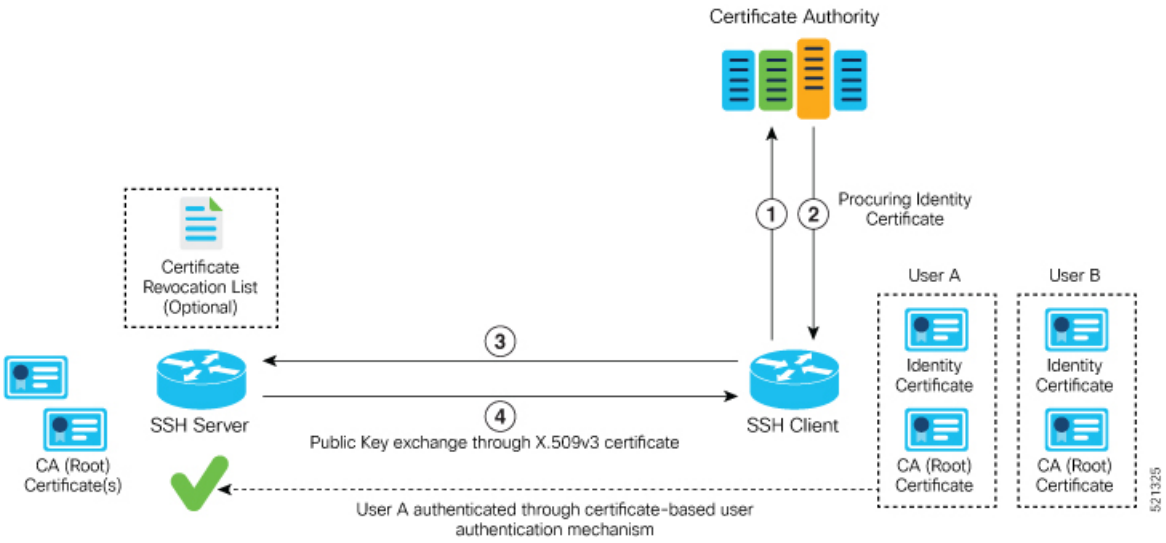
Server Authentication using X.509v3 Certificate



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

Perform this task to enable X.509v3 certificate-based server and user authentication for SSH.

Server Authentication:

- Configure the list of host key algorithms—With this configuration, the SSH server decides the list of host keys to be offered to the client. In the absence of this configuration, the SSH server sends all available algorithms to the user as host key algorithms. The SSH server sends these algorithms based on the availability of the key or the certificate.
- Configure the SSH trust point for server authentication—With this configuration, the SSH server uses the given trust point certificate for server authentication. In the absence of this configuration, the SSH server does not send **x509v3-ssh-rsa** as a method for server verification. This configuration is not VRF-specific; it is applicable to SSH running in all VRFs.

  The above two tasks are for server authentication and the following ones are for user authentication.

User Authentication:

- Configure the trust points for user authentication—With this configuration, the SSH server uses the given trust point for user authentication. This configuration is not user-specific; the configured trust points are used for all users. In the absence of this configuration, the SSH server does not authenticate using certificates. This configuration is not specific to a VRF; it is applicable to SSH running in all VRFs.

  You can configure up to ten user trust points.
- Specify the *username* to be picked up from the certificate—This configuration specifies which field in the certificate is to be considered as the *username*. The **common-name** from the **subject name** or the **user-principle-name(othername)** from the **subject alternate name**, or both can be configured.
- Specify the maximum number of authentication attempts allowed by the SSH server—The value ranges from 4 to 20. The default value is 20. The server closes the connection if the number of user attempts exceed the configured value.
- AAA authentication configuration—The AAA configuration for public key is the same as that for the regular or keyboard-interactive authentication, except that it mandates local method in the authentication method list.

Configuration Example

In this example, the **x509v3-ssh-rsa** is specified as the allowed host key algorithm to be sent to the client. Similarly, you can configure other algorithms, such as **ecdsa-sha2-nistp521, ecdsa-sha2-nistp384, ecdsa-sha2-nistp256, ssh-rsa**, and **ssh-dsa**.

```
/* Configure the lits of host key algorithms */
Router#configure
Router(config)#ssh server algorithms host-key x509v3-ssh-rsa
Router(config)#commit


/* Configure the SSH trustpoint for server authentication */
Router#configure
Router(config)#ssh server certificate trustpoint host tp1
Router(config)#commit


/* Configure the trustpoints to be used for user authentication */
Router#configure
Router(config)#ssh server trustpoint user tp1
Router(config)#ssh server trustpoint user tp2
Router(config)#commit


/* Specifies the username to be picked up from the certificate.
In this example, it specifies the user common name to be picked up from the subject name
field */
Router#configure
Router(config)#ssh server certificate username common-name
Router(config)#commit


/* Specifies the maximum authentication limit for the SSH server */
Router#configure
Router(config)#ssh server max-auth-limit 5
Router(config)#commit


/* AAA configuration for local authentication with certificate and
remote authorization with TACACS+ or RADIUS */
Router#configure
Router(config)#aaa authentication login default group tacacs+ local
Router(config)#aaa authorization exec default group radius group tacacs+
Router(config)#commit
```

Running Configuration

```
ssh server algorithms host-key x509v3-ssh-rsa
!
ssh server certificate trustpoint host tp1
!
ssh server trustpoint user tp1
ssh server trustpoint user tp2
!
ssh server certificate username common-name
!
ssh server max-auth-limit 5
!
```

Verification of Certificate-based Authentication for SSH

You can use the **show ssh server** command to see various parameters of the SSH server. For certificate-based authentication for SSH, the **Certificate Based** field displays *Yes*. Also, the two new fields, **Host Trustpoint** and **User Trustpoints**, display the respective trust point names.

```
Router#show ssh server
```

```
Wed Feb 19 15:23:38.752 IST
---------------------
SSH Server Parameters
---------------------

Current supported versions := v2
                 SSH port := 22
                 SSH vrfs := vrfname:=default(v4-acl:=, v6-acl:=)
             Netconf Port := 830
             Netconf Vrfs := vrfname:=default(v4-acl:=, v6-acl:=)

 Algorithms
---------------
        Hostkey Algorithms := x509v3-ssh-rsa,
ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,ssh-rsa,ssh-dsa
   Key-Exchange Algorithms :=
ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group14-sha1
     Encryption Algorithms :=
aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
            Mac Algorithms := hmac-sha2-512,hmac-sha2-256,hmac-sha1

 Authetication Method Supported
-----------------------------------
                PublicKey := Yes
                 Password := Yes
      Keyboard-Interactive := Yes
         Certificate Based := Yes

 Others
------------
                     DSCP  := 16
                Ratelimit  := 60
             Sessionlimit  := 100
                Rekeytime  := 60
       Server rekeyvolume  := 1024
  TCP window scale factor  := 1
            Backup Server  := Enabled, vrf:=default, port:=11000
Host Trustpoint            := tp1
User Trustpoints           := tp1 tp2
```

You can use the **show ssh session details** command to see the chosen algorithm for an SSH session:

```
Router#show ssh session details
Wed Feb 19 15:33:00.405 IST
SSH version : Cisco-2.0

id      key-exchange           pubkey               incipher    outcipher   inmac
outmac
-----------------------------------------------------------------------------------------------

Incoming Sessions
1       ecdh-sha2-nistp256     x509v3-ssh-rsa       aes128-ctr  aes128-ctr  hmac-sha2-256
hmac-sha2-256
```

Similarly, you can use the **show ssh** command to verify the authentication method used. In this example, it shows as *x509-rsa-pubkey*:

```
Router#show ssh
Sun Sep 20 18:14:04.122 UTC
SSH version : Cisco-2.0
```

```
id  chan pty  location   state        userid    host          ver authentication connection
 type
--------------------------------------------------------------------------------------------------
Incoming sessions
4   1    vty0 0/RP0/CPU0 SESSION_OPEN  9chainuser 10.105.230.198 v2  x509-rsa-pubkey
Command-Line-Interface

Outgoing sessions
```

SYSLOGS

You can observe relevant SYSLOGS on the router console in various scenarios listed here:

- On successful verification of peer certificate:

```
RP/0/RP0/CPU0:Aug 10 15:01:34.793 UTC: locald_DLRSC[133]: %SECURITY-PKI-6-LOG_INFO :
Peer certificate verified successfully
```

- When user certificate CA is not found in the trust point:

```
RP/0/RP0/CPU0:Aug  9 22:06:43.714 UTC: locald_DLRSC[260]: %SECURITY-PKI-3-ERR_GENERAL
: issuer not found in trustpoints configured
RP/0/RP0/CPU0:Aug  9 22:06:43.714 UTC: locald_DLRSC[260]: %SECURITY-PKI-3-ERR_ERRNO :
Error:='Crypto Engine' detected the 'warning' condition 'Invalid trustpoint or trustpoint
 not exist'(0x4214c000), cert verificationn failed
```

- When there is no CA certificate or host certificate in the trust point:

```
RP/0/RP1/CPU0:Aug 10 00:23:28.053 IST: SSHD_[69552]: %SECURITY-SSHD-4-WARNING_X509 :
could not get the host cert chain, 'sysdb' detected the 'warning' condition 'A SysDB
client tried to access a nonexistent item or list an empty directory', x509 host auth
will not be used
RP/0/RP1/CPU0:Aug 10 00:23:30.442 IST: locald_DLRSC[326]: %SECURITY-PKI-3-ERR_ERRNO :
Error:='Crypto Engine' detected the 'warning' condition 'Invalid trustpoint or trustpoint
 not exist'(0x4214c000), Failed to get trustpoint name from
```

How to Disable X.509v3 Certificate-based Authentication for SSH

- Server Authentication — You can disable X.509v3 certificate-based server authentication for SSH by using the **ssh server algorithms host-key** command. From the list of auto-generated host-key pairs algorithms on the SSH server, this command configures allowed SSH host-key pair algorithms. Hence, if you have this configuration without specifying the **x509-ssh-rsa** option in the preceding command, it is equivalent to disabling the X.509v3 certificate-based server authentication for the SSH server.
- User Authentication — You can remove the user trust point configuration (**ssh server trustpoint user**) so that the SSH server does not allow the X.509v3 certificate-based authentication.

Failure Modes for X.509v3 Certificate-based Authentication for SSH

If the **ssh server certificate trustpoint host** configuration is missing, or if the configuration is present, but the router certificate is not present under the trust point, then the SSH server does not add **x509-ssh-rsa** to the list of supported host key methods during key exchange.

Also, the user authentication fails with an error message if:

- User certificate is in an incorrect format.

- The chain length of the user certificate is more than the maximum limit of 9.
- Certificate verification fails due to any reason.

Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)	Release 7.9.1	With this feature, the router can handle the X.509v3 Certificates Extensions defined in RFC 5280 while validating the client certificate over mTLS. Here, the router acknowledges all extensions in X.509v3 Certificates of the user while validating it. Previously, the router failed to process certification extensions when the severity was critical and resulting in authentication failure. This feature permits users to configure any certificate extensions with different severity in their X.509v3 Certificates.

Starting with IOS XR Release 7.9.1, you can add any Certificate Extensions available in RFC 5280 to your X.509v3 client certificates validations over Mutual Transport Layer Security (mTLS). While validating such client certificate, the router acknowledges all extensions available in the certificate presented and processes it. With this, the router allows the user to configure any number of extensions with different severity in their X.509v3 client certificates.

Related Topics

- X.509v3 Certificate-based Authentication for SSH, on page 278

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**

Selective Authentication Methods for SSH Server

Table 44: Feature History Table

Feature Name	Release Information	Feature Description
Selective Authentication Methods for SSH Server	Release 7.8.1	You now have the flexibility to choose the preferred SSH server authentication methods on the router. These methods include password authentication, keyboard-interactive authentication, and public-key authentication. This feature allows you to selectively disable these authentication methods. By allowing the SSH clients to connect to the server only through these permitted authentication methods, this functionality brings in additional security for router access through SSH. Before this release, by default, the SSH server allowed all these authentication methods for establishing SSH connections. The feature introduces these changes: • **CLI**: New **disable auth-methods** command • **YANG Data Model**: New XPaths for `Cisco-IOS-XR-crypto-ssh-cfg.yang` Cisco native model (see GitHub)

By default, the SSH server on the Cisco IOS XR routers allowed various authentication methods such as password authentication, keyboard-interactive authentication, and public-key authentication (including certificate-based authentication) for the SSH connections on the router. The SSH clients could use any of these authentication methods while attempting a connection to the SSH server on the router. From Cisco IOS XR Software Release 7.8.1, you can selectively disable these authentication methods, and allow connection attempts from the SSH client only through the remaining authentication methods. If the SSH client tries to establish a connection to the server using nonpermitted authentication methods (the ones that are disabled), then the login attempt fails.

Disable SSH Server Authentication Methods

Use the **disable auth-methods** command in ssh server configuration mode to disable the specific authentication method for the SSH server.

Public-key authentication includes certificate-based authentication as well. Hence, disabling public-key authentication automatically disables the certificate-based authentication.

Configuration Example

This example shows how to disable the keyboard-interactive authentication method for the SSH server on the router using CLI. Similarly, you can disable other authentication methods.

```
Router#configure
Router(config)# ssh server
Router(config-ssh)# disable auth-methods keyboard-interactive
Router(config-ssh)# commit
```

Running Configuration

```
!
ssh server
 disable auth-methods keyboard-interactive
!
```

Verification

Use the **show ssh server** command to see the list of authentication methods that the SSH server on the router supports. In this example, the keyboard-interactive method is disabled and the SSH server allows all other authentication methods.

```
Router#show ssh server

Wed Feb 23 10:38:37.716 UTC
Authentication Method Supported
-----------------------------------
                 PublicKey := Yes
                  Password := Yes
      Keyboard-Interactive := No
         Certificate Based := Yes
```

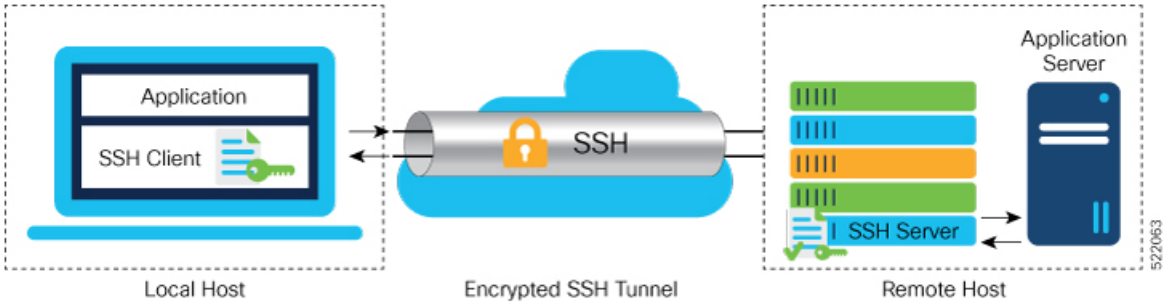
SSH Port Forwarding

Table 45: Feature History Table

Feature Name	Release Information	Feature Description
SSH Port Forwarding with CiscoSSH	Release 7.3.2	This release introduces SSH port forwarding with CiscoSSH, an OpenSSH-based implementation of SSH. CiscoSSH replaces Cisco IOS XR SSH, which is the older SSH implementation that existed prior to this release.
SSH Port Forwarding with Cisco IOS XR SSH	Release 7.3.15	With this feature enabled, the SSH client on a local host forwards the traffic coming on a given port to the specified host and port on a remote server, through an encrypted SSH channel. Legacy applications that do not otherwise support data encryption can leverage this functionality to ensure network security and confidentiality to the traffic that is sent to remote application servers. This feature introduces the ssh server port-forwarding local command.

SSH port forwarding or SSH tunneling is a method of forwarding the otherwise insecure TCP/IP connections from the SSH client to server, or the other way around, 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 15: SSH Port Forwarding

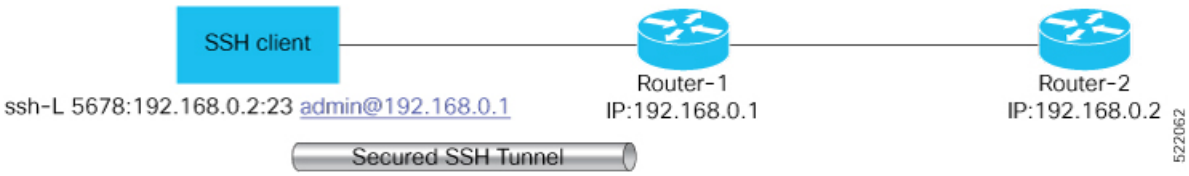


Consider an application on the SSH client residing on a local host, trying to connect to an application server residing on a remote host. The remote host can either be a single router where both the SSH server and application server reside, or, it can host the SSH server on one router and application server on a different device, like in case of a data center. With port forwarding or 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 16: 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 port 23, in this example.

sshserver-hostname is the domain name or IP address of the SSH server that receives the SSH client request. It must the SSH IP address or domain name to access the router that hosts the SSH server. That is, 192.168.0.1 of Router-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 application server 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 **port-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
Tue Sep  7 17:43:22.483 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,rsa-sha2-512,rsa-sha2-256,ssh-rsa,ssh-dsa,ssh-ed25519

   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

 Authentication Method Supported
------------------------------------
                 PublicKey := Yes
                  Password := Yes
      Keyboard-Interactive := Yes
         Certificate Based := Yes

 Others
------------
                     DSCP  := 0
                Ratelimit  := 600
             Sessionlimit  := 110
                Rekeytime  := 30
       Server rekeyvolume  := 1024
  TCP window scale factor  := 1
```

```
            Backup Server  := Disabled
          Host Trustpoint  :=
          User Trustpoint  := tes,test,x509user
          Port Forwarding  := local
Max Authentication Limit  := 16
     Certificate username  := Common name(CN) User principle name(UPN)
Router#
```

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:

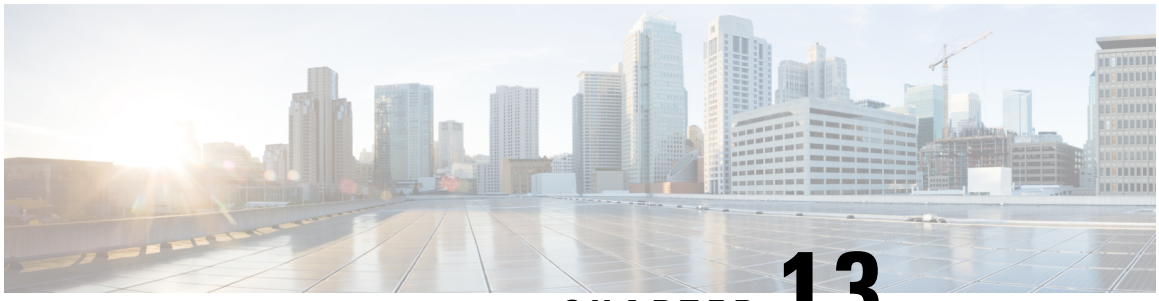
```
RP/0/RP0/CPU0:Aug 24 13:10:15.933 IST: SSHD_[66632]:
%SECURITY-SSHD-6-PORT_FWD_INFO_GENERAL : Port Forwarding, Target:=10.105.236.155,
Port:=22, Originator:=127.0.0.1,Port:=41590, Vrf:=0x60000000, Connection forwarded
```

- 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 13

Configuring FIPS Mode

The Federal Information Processing Standard (FIPS) 140-2 is an U.S. and Canadian government certification standard that defines requirements that the cryptographic modules must follow. The FIPS specifies best practices for implementing cryptographic algorithms, handling key material and data buffers, and working with the operating system.

In Cisco IOS XR software, these applications are verified for FIPS compliance:

- Secure Shell (SSH)
- Secure Socket Layer (SSL)
- Transport Layer Security (TLS)
- Internet Protocol Security (IPSec) for Open Shortest Path First version 3 (OSPFv3)
- Simple Network Management Protocol version 3 (SNMPv3)
- AAA Password Security

Note Any process that uses any of the following cryptographic algorithms is considered non-FIPS compliant:

- Rivest Cipher 4 (RC4)
- Message Digest (MD5)
- Keyed-Hash Message Authentication Code (HMAC) MD5
- Data Encryption Standard (DES)

The Cisco Common Cryptographic Module (C3M) provides cryptographic services to a wide range of the networking and collaboration products of Cisco. This module provides FIPS-validated cryptographic algorithms for services such as RTP, SSH, TLS, 802.1x, and so on. The C3M provides cryptographic primitives and functions for the users to develop any protocol.

By integrating with C3M, the Cisco IOS-XR software is compliant with the FIPS 140-2 standards and can operate in FIPS mode, level 1 compliance.

- Prerequisites for Configuring FIPS, on page 294
- How to Configure FIPS, on page 294

Prerequisites for Configuring FIPS

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.

Guidelines for Enabling FIPS Mode

From Cisco IOS XR Software Release 7.2.1 and later, you must follow these guidelines while enabling FIPS mode:

- 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, and only for FIPS mode (that is, when **crypto fips-mode** is configured).
- 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. This is applicable only for FIPS mode.
- If you try to execute the telnet configuration on a system where the FIPS mode is already enabled, then the system rejects the telnet configuration.
- If telnet configuration already exists on the system, and if FIPS mode is enabled later, then the system rejects the telnet connection. But, it does not affect the telnet configuration as such.
- It is recommended to configure the **crypto fips-mode** command first, followed by the commands related to FIPS in a separate commit. The list of commands related to FIPS with non-approved cryptographic algorithms are:
  - **key chain** *key-chain-name* **key** *key-id* **cryptographic-algorithm MD5**
  - **key chain** *key-chain-name* **key** *key-id* **cryptographic-algorithm HMAC-MD5**
  - **router ospfv3 1 authentication ipsec spi 256 md5** *test-md5-value*
  - **router ospfv3 1 encryption ipsec spi 256 esp des** *test-des-value*
  - **router ospfv3 1 encryption ipsec spi 256 esp des** *test-des-value* **authentication md5** *test-md5-value*
  - **snmp-server user** *user1 user-grp1* **v3 auth md5 priv des56**
  - **ssh server algorithms key-exchange diffie-hellman-group1-sha1**
  - **telnet vrf default ipv4 server max-servers** *100*

How to Configure FIPS

Perform these tasks to configure FIPS.

Enable FIPS mode

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **crypto fips-mode**

Example:

```
Router(config)#crypto fips-mode
```

Enters FIPS configuration mode.

Note
Stop new incoming SSH sessions while configuring or unconfiguring **crypto fips-mode**. Restart the router upon configuration.

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 logging**

Example:

```
Router#show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 60 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 3 messages logged

Log Buffer (9000000 bytes):
<output omitted>

Log Buffer (307200 bytes):

RP/0/RSP0/CPU0:Apr 16 12:48:17.736 : cepki[433]: The configuration setting for FIPS mode has been
modified. The system must be reloaded to finalize this configuration change. Please refer to the IOS
 XR System Security Configuration Guide, Federal Information Process Standard(FIPS) Overview section
 when modifying the FIPS mode setting.
RP/0/RSP0/CPU0:Apr 16 12:48:17.951 : config[65757]: %MGBL-CONFIG-6-DB_COMMIT :
```

```
Configuration committed by user 'lab'. Use 'show configuration commit changes 1000000002' to view
the changes.
RP/0/RSP0/CPU0:Apr 16 12:48:23.988 : config[65757]: %MGBL-SYS-5-CONFIG_I : Configured from console
by lab


....
....
....
```

Displays the contents of logging buffers.

Note
Use the **show logging | i fips** command to filter FIPS specific logging messages.

Step 5 **reload location all**

Example:

```
Router#reload location all
```

Reloads a node or all nodes on a single chassis or multishelf system.

Configure FIPS-compliant Keys

Perform these steps to configure the FIPS-compliant keys:

Note The crypto keys are auto-generated at the time of router boot up. You need to perform these steps to generate the keys only if the keys are missing under some scenarios.

Note From Cisco IOS XR Release 7.3.2 onwards, you can generate and delete key-pairs from XR Config mode. For more details, see Public Key-Pair Generation in XR Config Mode, on page 144 in the chapter *Implementing Certification Authority Interoperability* .

Before you begin

Refer the configuration steps in the Enable FIPS mode, on page 295 section for enabling the FIPS mode.

Procedure

Step 1 **crypto key generate rsa [usage-keys | general-keys]** *key label*

Example:

```
Router#crypto key generate rsa general-keys rsakeypair
```

Generate a RSA key pair. Ensure that all the key pairs meet the FIPS requirements. The RSA key sizes allowed under FIPS mode are 2048, 3072 and 4096.

The option **usage-keys** generates separate RSA key pairs for signing and encryption. The option **general-keys** generates a general-purpose RSA key pair for signing and encryption.

To delete the RSA key pair, use the **crypto key zeroize rsa** *keypair-label* command.

Step 2 **crypto key generate dsa**

Example:

```
Router#crypto key generate dsa
```

Generate a DSA key pair if required. Ensure that all the key pairs meet the FIPS requirements. The DSA key size allowed under FIPS mode is 2048.

To delete the DSA key pair, use the **crypto key zeroize dsa** *keypair-label* command.

Step 3 **crypto key generate ecdsa**

Example:

```
Router#crypto key generate ecdsa
```

Generate a ECDSA key pair if required. Ensure that all the key pairs meet the FIPS requirements. The ECDSA key sizes allowed under FIPS mode are **nistp256**, **nistp384** and **nistp512**.

To delete the DSA key pair, use the **crypto key zeroize ecdsa** *keypair-label* command.

Step 4 **show crypto key mypubkey rsa**

Example:

```
Router# show crypto key mypubkey rsa
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
```

Displays the existing RSA key pairs.

You can also view the RSA keys in the running configuration. The keys in the following example are in OpenSSL format.

Note
Only those keys that are generated in the `config` mode are visible in the running configuration.

```
Router(config)#crypto key generate rsa test
Router(config)#commit
Thu May 12 08:37:59.894 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nTbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOTycjsizI/TLBz9WoQX.rmxVwkNgTKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

Step 5 **show crypto key mypubkey dsa**

Example:

```
Router#show crypto key mypubkey dsa
```

Displays the existing DSA key pairs.

You can also view the DSA key in the running configuration. The keys in the following example are in OpenSSL format:

Note
Only those keys that are generated in the `config` mode are visible in the running configuration.

```
Router(config)#crypto key generate dsa
Router(config)#commit
Thu May 12 08:39:32.093 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:39:37.557 UTC
```

```
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:39:32 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nTbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOTycjsizI/TLBz9WoQX.rmxVwkNgTKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate dsa 1024 | -----BEGIN PUBLIC KEY-----
MIIBcTCB8AYHKoZIzjgEATCB5AKBgQCPJcWeV38AANyKXp5XfwAAjyXFnld/AADI
CQDQVn8AAADt274AAAAA2hu9QE4nZsllAAAAAAAAADAXANBWfwAA3Ipenld/AAAg
AAAAAAAAAIIlxZ5XfwAA3Ipenld/AACCJcWeV38AACtah6B+cCkKAO3bvgAAAADc
gpej73WrUwIUSD4A0FZ/AAAQAAAAEAAAAAAAAAACSAEAAAAAAAAAAAAAAAAAAAAB
AAAAAAAAAAAAAAAAAAAAAYJtQfFd/AAAAAAAAAAAAACgzANBWfwAAAO3bvgAAAAC2
3xsW4Kjy1wN8AAJ5EAAAABAAAH9W0AA+SAAAAAAA7du+AAAAAFUAAAAAAAAAoDEA
0FZ/AAAV8DieV38AACAAAAAAAAAAHvE4nld/AAAV8DieV38AAB7xOJ5XfwAAAAAA
AAAAAAAA7du+AAAAAJEOuuQWSrUHVQAAAAAAAAAAAAAAAAAAAAAA==
-----END PUBLIC KEY-----
 |
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end|
```

Configure FIPS-compliant Key Chain

Perform these steps to configure the FIPS-compliant key chain:

Before you begin

Refer the configuration steps in the Enable FIPS mode, on page 295 section for enabling the FIPS mode.

Procedure

Step 1 **configure**

Example:

```
Router#configure
```

Enters the global configuration mode.

Step 2 **key chain** *key-chain-name*

Example:

```
Router(config)#key chain mykeychain
```

Creates a key chain.

Step 3 **key** *key-id*

Example:

```
Router(config-mykeychain)#key 1
```

Creates a key in the key chain.

Step 4 **cryptographic-algorithm** {**HMAC-SHA1-20** | **SHA-1**}

Example:

```
Router(config-mykeychain-1)#cryptographic-algorithm HMAC-SHA1-20
```

Configures the cryptographic algorithm for the key chain. Ensure that the key chain configuration always uses SHA-1 as the hash or keyed hash message authentication code (hmac) algorithm.

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 FIPS-compliant Certificates

Perform these steps to configure the FIPS-compliant certificates:

Before you begin

Refer the configuration steps in the Enable FIPS mode, on page 295 section for enabling the FIPS mode.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **crypto ca trustpoint** *ca-name key label*

Example:

```
Router(config)#crypto ca trustpoint msiox rsakeypair
```

Configures the trustpoint by utilizing the desired RSA keys.

Ensure that the certificates meet the FIPS requirements for key length and signature hash or encryption type.

Note
The minimum key length for RSA or DSA key is 1024 bits. The required hash algorithm is SHA-1-20.

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 crypto ca certificates**

Example:

```
Router#show crypto ca certificates
```

Displays the information about the certificate

What to do next

For more information about certification authority and requesting router certificates, see the *Implementing Certification Authority* chapter in this guide.

Configure FIPS-compliant OSPFv3

Perform these steps to configure the FIPS-compliant OSPFv3:

Before you begin

Refer the configuration steps in the Enable FIPS mode, on page 295 section for enabling the FIPS mode.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospfv3** *process name*

Example:

```
Router(config)#router ospfv3 ospfname
```

Configures the OSPFv3 process.

Step 3 **area** *id*

Example:

```
Router(config-ospfv3)#area 1
```

Configures the OSPFv3 area ID. The ID can either be a decimal value or an IP address.

Step 4 **authentication** {**disable** | **ipsec spi** *spi-value* **sha1** [**clear** | **password**] *password*}

Example:

```
Router(config-ospfv3-ar)#authentication ipsec spi 256 sha1 password pa1
```

Enables authentication for OSPFv3. Note that the OSPFv3 configuration supports only SHA-1 for authentication.

Note
IPSec is supported only for Open Shortest Path First version 3 (OSPFv3).

Step 5 **exit**

Example:

```
Router(config-ospfv3-ar)#exit
```

Exits OSPFv3 area configuration and enters the OSPFv3 configuration mode.

Step 6 **encryption** {**disable** | {**ipsec spi** *spi-value* **esp** {**3des** | **aes** [**192** | **256**] [**clear** |**password**] *encrypt-password*} [**authentication sha1**[**clear** | **password**] *auth-password*]}}

Example:

```
Router(config-ospfv3)#encryption ipsec spi 256 esp 3des password pwd
```

Encrypts and authenticates the OSPFv3 packets. Ensure that the OSPFv3 configuration uses the following for encryption in the configuration.

- 3DES: Specifies the triple DES algorithm.
- AES: Specifies the Advanced Encryption Standard (AES) algorithm.

Ensure that SHA1 is chosen if the authentication option is specified.

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 FIPS-compliant SNMPv3 Server

Perform these steps to configure the FIPS-compliant SNMPv3 server:

Before you begin

Refer the configuration steps in the Enable FIPS mode, on page 295 section for enabling the FIPS mode.

Procedure

Step 1 **configure**

Example:

```
Router#configure
```

Enters the global configuration mode.

Step 2 **snmp-server user** *username groupname* {**v3** [**auth sha** {**clear** | **encrypted**} *auth-password* [**priv** {**3des** |**aes** { **128** | **192** | **256**} } {**clear** | **encrypted** } *priv-password*]] } [**SDROwner** | **SystemOwner**] *access-list-name*

Example:

```
Router(config)#snmp-server user user1 g v3 auth sha clear pass priv aes 128 clear privp
```

Configures the SNMPv3 server.

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.

Configure FIPS-compliant SSH Client and Server

Perform these steps to configure the FIPS-compliant SSH Client and the Server:

Before you begin

Refer the configuration steps in the Enable FIPS mode, on page 295 section for enabling the FIPS mode.

Procedure

Step 1 **ssh** {*ipv4-address* | *ipv6-address*} **cipher aes** {**128-CTR** | **192-CTR** | **256-CTR**} **username** *username*

Example:

```
Router#ssh 192.0.2.1 cipher aes 128-CTR username user1
```

Starts an SSH session to the server using the FIPS-approved ciphers. Ensure that the SSH client is configured only with the FIPS-approved ciphers. AES(Advanced Encryption Standard)-CTR (Counter mode) is the FIPS-compliant cipher algorithm with key lengths of 128, 192 and 256 bits.

Step 2 **configure**

Example:

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

Enters mode.

Step 3 **ssh server v2**

Example:

```
Router(config)#ssh server v2
```

Configures the SSH server.

The supported key exchange algorithms are:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The supported cipher algorithms are:

- aes128-ctr
- aes192-ctr
- aes256-ctr
- aes128-gcm
- aes256-gcm

The supported HMAC algorithms are:

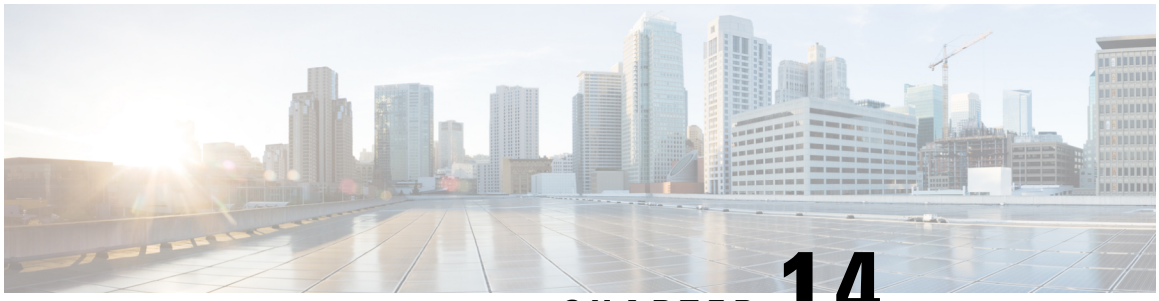
- hmac-sha2-512
- hmac-sha2-256
- hmac-sha1

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.



CHAPTER 14

Implementing Secure Logging

This chapter describes the implementation of secure logging over Transport Layer Security (TLS). TLS, the successor of Secure Socket Layer (SSL), is an encryption protocol designed for data security over networks.

Table 46: Feature History Table

Release	Modification
Release 7.0.12	This feature was introduced.

- System Logging over Transport Layer Security (TLS), on page 307
- Restrictions for Syslogs over TLS, on page 309
- Configuring Syslogs over TLS, on page 309

System Logging over Transport Layer Security (TLS)

System Log (syslog) messages indicate the health of the device and provide valuable information about any problems encountered. By default, the syslog process sends messages to the console terminal.

Due to limited size of the logging buffer in a router, these syslog messages get overwritten in a short time. Moreover, the logging buffer doesn't retain syslogs across router reboots. To avoid these issues, you can configure the router to send syslog messages to an external syslog server for storage.

Note For more information on configuring system logging, see *Implementing System Logging* chapter in the *System Monitoring Configuration Guide for Cisco 8000 Series Routers*

Traditionally, routers transfer syslogs to an external syslog server using User Datagram Protocol (UDP), which is an insecure way of transferring logs. To guarantee secure transport of syslogs, the Cisco 8000 Series Router supports Secure Logging based on RFC 5425 (Transport Layer Security Transport Mapping for Syslog). With this feature, the router sends syslogs to a remote server, over a trusted channel which implements the secure Transport Layer Security (TLS) encryption protocol.

TLS ensures secure transport of syslogs by:

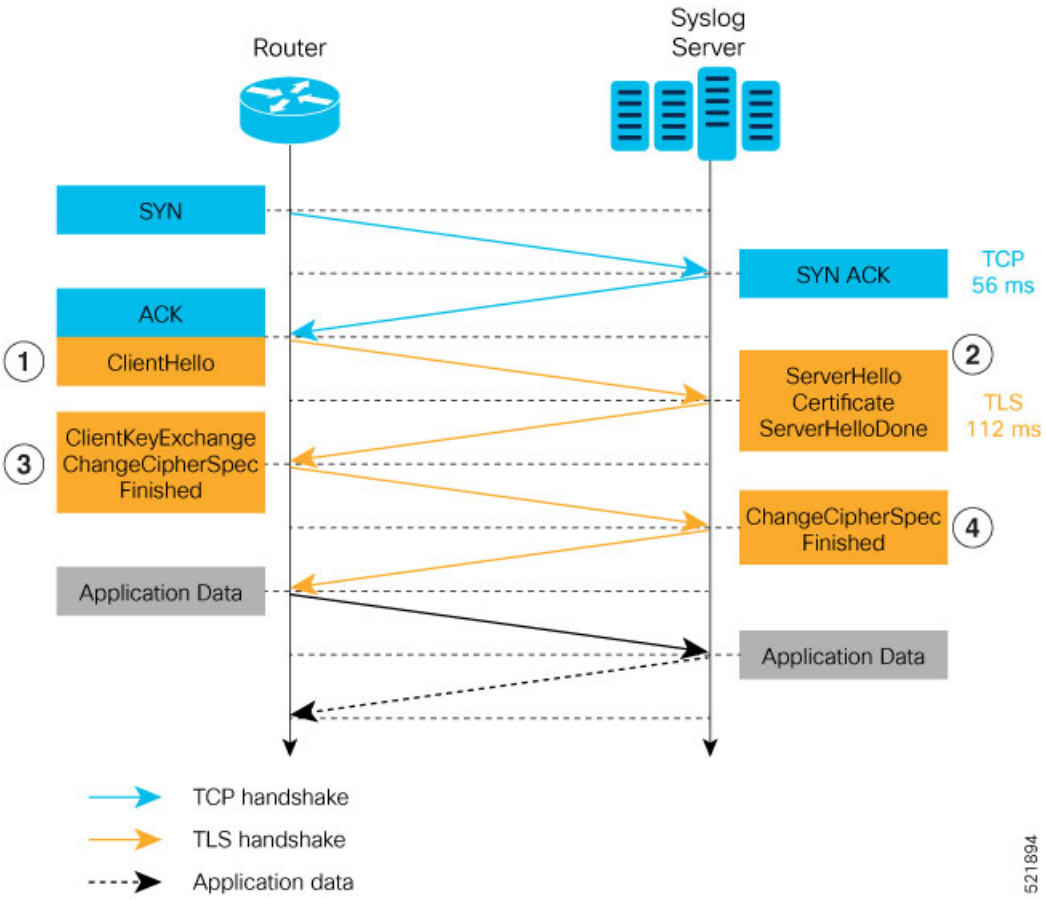
- Authenticating the server and client
- Encrypting the syslog data transferred

- Verifying the integrity of data

The router is the TLS client and remote syslog server is the TLS server. TLS runs over Transmission Control Protocol (TCP). So, the client must complete the TCP handshake with the server before starting TLS handshake.

Sequence of TLS Handshake

Figure 17: TLS Handshake



To establish the TLS session, the following interactions take place between the router and the syslog server after TCP handshake is complete:

1. The router sends Client Hello message to the server to begin TLS handshake.
2. The server shares its TLS certificate, which contains its public key, with the router to establish a secure connection.
3. The router confirms the server certificate with the Certification Authority and checks the validity of the TLS certificate. Then, the router sends a Change Cipher Spec message to the server to indicate that messages sent are encrypted using the negotiated key and algorithm.
4. The server decrypts the message using its private key. And then, sends back a Change Cipher Spec message encrypted with the session key to complete the TLS handshake and establish the session.

For more information on configuring Certification Authority interoperability, refer *Implementing Certification Authority Interoperability* chapter in this guide.

Restrictions for Syslogs over TLS

The following restrictions apply for sending syslogs to a remote syslog server over TLS:

- While configuring the settings for the syslog server on the router, specify only one server identifier, either the hostname or the ipv4/v6 address.
- In the TLS certificate of the syslog server, if Subject Alternative Name (SAN) field matches the configured server hostname but Common Name (CN) field doesn't match the configured server hostname, TLS session setup fails.

Configuring Syslogs over TLS

The following steps show how to configure syslog over TLS:

1. Configure the trust-point for establishing the TLS channel as shown:

```
Router#conf t
Router(config)#crypto ca trustpoint tp
Router(config-trustp)#subject-name CN=new
Router(config-trustp)#enrollment terminal
Router(config-trustp)#rsakeypair k1
Router(config-trustp)#commit
```

Note You can either use the command **enrollment url** *SCEP-url* or the command **enrollment terminal** for configuring trustpoint certification authority (CA) enrollment. For more information, see *Implementing Certification Authority Interoperability* chapter in this guide.

2. Configure the settings to access the remote syslog server. You can use either the IPv4/v6 address of the server or the server hostname for this configuration. Based on the configured **severity**, the router sends syslogs to the server. Logging severity options include **alerts, critical, debugging, emergencies, errors, informational, notifications and warnings**. For more information about logging severity levels, see the topic *Syslog Message Severity Levels* in *Implementing System Logging* chapter in *System Monitoring Configuration Guide for Cisco 8000 Series Routers*.

   This example shows you how to configure syslog server settings with the IPv4 address.

```
Router(config)#logging tls-server TEST
Router(config-logging-tls-peer)#severity debugging
Router(config-logging-tls-peer)#trustpoint tp
Router(config-logging-tls-peer)#address ipv4 10.105.230.83
Router(config-logging-tls-peer)#commit
```

   Alternately, you can configure the syslog server settings with server hostname instead of the IPv4/v6 address.

```
Router(config)#logging tls-server TEST
Router(config-logging-tls-peer)#severity debugging
Router(config-logging-tls-peer)#trustpoint tp
```

```
Router(config-logging-tls-peer)#tls-hostname xyz.cisco.com
Router(config-logging-tls-peer)#commit
```

3. Configure the domain to map the IP address of the remote syslog server and its hostname.

```
Router(config)#domain ipv4 host xyz.cisco.com 10.105.230.83
Router(config)#domain name cisco.com
Router(config)#commit
```

Verification Steps

TCP port 6514 is the default port for syslog over TLS. Verify the TLS configuration by checking if port 6514 is associated with the IP address of the syslog server in the output of the command **show lpts bindings brief**.

```
Router#show lpts bindings brief

@ - Indirect binding; Sc - Scope

 Location   Clnt Sc L3   L4    VRF-ID    Interface    Local-Address,Port Remote-Address,Port

 ---------- ---- -- ---- ------ --------- ------------ ------------------------------------

  0/RP0/CPU0 TCP LR IPV4 TCP    default   any          5.10.18.5,35926 10.105.230.83,6514
```

The output of **show logging** command displays the IP address of the TLS server and the number of messages sent to the remote syslog server.

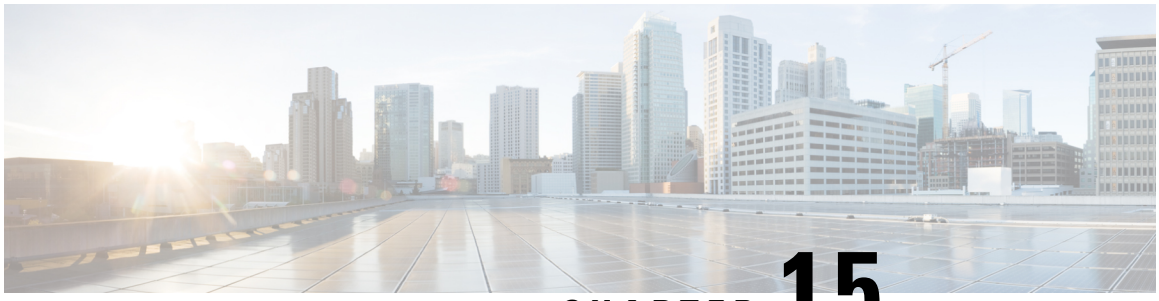
```
Router#show logging

Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 185 messages logged
    Monitor logging: level debugging, 94 messages logged
    Trap logging: level informational, 0 messages logged
    Logging to TLS server 10.105.230.83, 66 message lines logged
    Buffer logging: level debugging, 183 messages logged

Log Buffer (2097152 bytes):
...................
```

The output of **show crypto ca certificates** command displays the Certification Authority (CA) certificate details.

```
Router#show crypto ca certificates

Trustpoint       : tp
================================================
CA certificate
  Serial Number  : B5:68:C8:96:A4:7C:1A:BA
  Subject:
      CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
      CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 05:39:51 UTC Tue Aug 13 2019
  Validity End   : 05:39:51 UTC Mon Aug 08 2039

  CRL Distribution Point
      http://10.105.236.78/crl_xxx/crl.der
  SHA1 Fingerprint:
      03BD57E04A2AA4648A84F515A46EF99CCF488387
```

When the TLS channel between the router and syslog server comes up, the router displays the following syslog messages on the console:

```
RP/0/RP0/CPU0: syslogd[148]: %SECURITY-XR_SSL-6-CERT_VERIFY_INFO : SSL Certificate
verification: Peer certificate verified successfully
RP/0/RP0/CPU0: syslogd[148]: %OS-SYSLOG-5-LOG_NOTICE : Secure Logging: Successfully
established TLS session , server :10.105.230.83
```



CHAPTER 15

Implementing MAC Authentication Bypass

This chapter describes the implementation of MAC Authentication Bypass (MAB).

IEEE 802.1X authentication configuration on the router helps to prevent unauthorized end devices from gaining access to the network. However, not all end devices support 802.1X. Hence, we introduce port controlling functionality on these routers using MAC authentication bypass (MAB)—a feature that grants network access to devices based on their MAC addresses, regardless of their 802.1X capability or credentials.

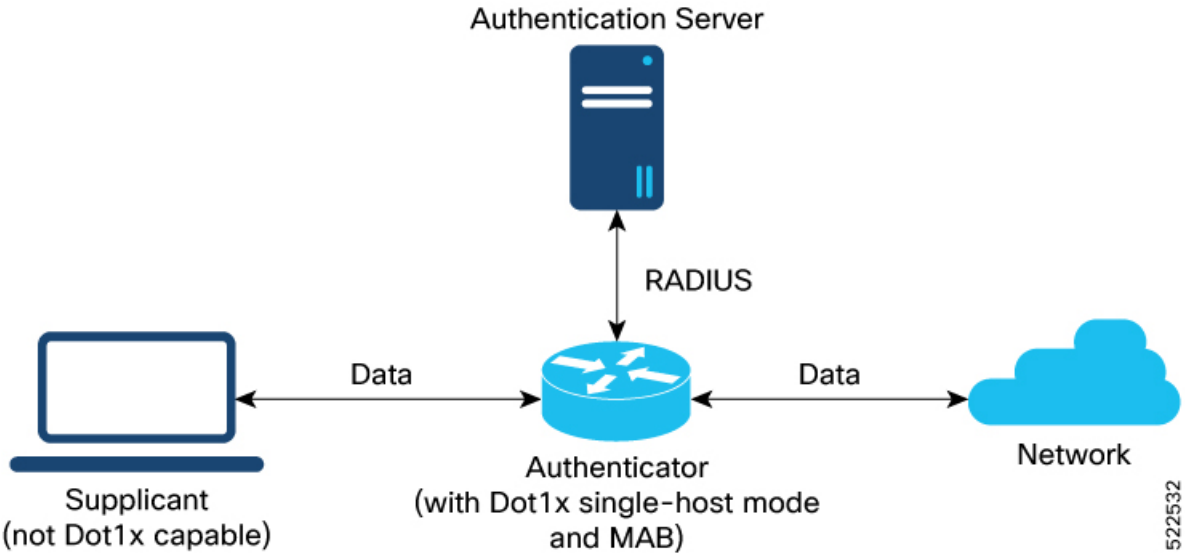
For details of commands related to MAB, see the *802.1X and Port Control Commands* chapter in the *System Security Command Reference for Cisco 8000 Series Routers*.

- MAC Authentication Bypass, on page 314

MAC Authentication Bypass

Table 47: Feature History Table

Feature Name	Release Information	Feature Description
MAC Authentication Bypass	Release 7.3.4 Release 7.5.2	Based on the MAC address of the end device or the client connected to the router port, this feature enables port control functionality for your router. This functionality provides controlled access to network services for end devices that do not support other authentication methods such as IEEE 802.1X port-based authentication. The MAB support is only for the **single-host** mode. This feature introduces these commands and options: • **mab** option in the **dot1x profile** command • **mab-retry-time** option in the **authenticator** command • **clear mab** • **show mab**



With MAC authentication bypass (MAB) functionality, the router (authenticator) uses the MAC address of the end device or the client (also called as supplicant) as an authenticating parameter for providing network access. With MAB enabled, when the router receives an incoming data packet from the client that is connected to the router port, it learns the source MAC address and sends it to the external RADIUS server (authentication server) for authentication. The RADIUS authentication server maintains a database of MAC addresses for devices that require access to the network. Based on the authentication result, the router allows or drops the data packets from that client. If the RADIUS server returns a success (*Access-Accept*) message, it indicates that the MAC address is authenticated and the client is authorized to send traffic through that port. The router then programs that MAC address on the port to which the client is connected. The router allows the traffic from the client to be forwarded to the network. Similarly, if the RADIUS server returns a failure (*Access-Reject*) message, it indicates that the MAC address is unauthenticated. And hence the router drops further data packets from that client. Thus, the MAB feature brings in port control functionality for Cisco 8000 Series Routers and provides end devices a controlled access to network services.

Starting with Cisco IOS-XR Release 24.4.1, MAC Authentication Bypass (MAB) can now have multiple hosts by allowing MAC addresses on a single port, each authenticated separately. The router achieves this functionality by increasing the maximum limit on MAC learning capability from 1 to 2 clients. With this new ability, when **multi-auth** mode is configured under MAB, the router continues MAC -learning on a port after authenticating a client using MAB, until the second client authentication is begun. With this you can use MAB for multi-domain authentication by allowing two endpoints to be authenticated and managed separately on the same port.

Authentication Failure Scenarios with MAB

This table lists various authentication failure scenarios and the expected feature behavior with MAB:

Table 48: Authentication Failure Scenarios with MAB

Authentication Failure Scenarios	Expected MAB Feature Behavior
RADIUS server rejects the authentication request	• The router deletes the client programming on the port (if that client was already authenticated). • Router retries the authentication process twice with the RADIUS server at an interval of 60 seconds, by default. You can configure this interval using the **authenticator timer mab-retry-time** command. • If the server still does not authorize the client, then the router clears the client session and its programming on the port. • The router puts the port back in MAC learning mode to relearn a new MAC address.

Authentication Failure Scenarios	Expected MAB Feature Behavior
RADIUS server is not reachable during authentication process	With **server dead action auth-retry** command configured: • The router retains the programing of the client that was already authenticated. Else, the router deletes it. • Router retries the authentication process with the RADIUS server at an interval of 60 seconds until the server becomes available. You can configure this interval using the **authenticator timer mab-retry-time** command. • The router does not attempt to learn any new MAC address on the port. • To clear the client session and its programming on the router, you must use the **clear mab session** command. • The router puts the port back in MAC learning mode to relearn a new MAC address. Similarly, for an unauthenticated client, if the authentication does not happen after the retries, the router deletes the client context and puts the port back in MAC learning mode. Without **server dead action auth-retry** command configuration: • The router deletes the programming of the client that was already authenticated and retries authentication (as mentioned earlier). • If the client is still not authenticated, the router automatically clears the client session. • The router puts the port back in MAC learning mode to relearn a new MAC address.

Restrictions for MAB

The restrictions apply to the MAB feature:

- With MAB, user authentication can only be done using a remote AAA server; not using the local AAA server on the router.
- MAB feature works only as a standalone feature; not as a fallback mechanism for any other type of authentication failures.
- MAB supports only a single end device on each port. Hence, you must configure the authenticator (the router) to be in **single-host** mode. Starting with Cisco IOS-XR Release 24.4.1, you can configure the authenticator (the router) to be in **single-host** or **multi-auth** mode.

Configure MAC Authentication Bypass

Prerequisites

- Configure the remote RADIUS server (using the **radius-server** command), and authentication method with the RADIUS server (using the **aaa authentication dot1x** command) in

- Configure the 802.1X profile (using the **dot1x profile** command in XR Config mode)
- Configure the authenticator (using the **authenticator** command in dot1x profile configuration sub mode) with respective parameters such as:
  - Re-authentication time—**reauth-time**
  - Host mode—as **single-host**
  - Retry action for server-unreachable scenarios—**auth-retry** or **auth-fail**

See the *MACSec Using EAP-TLS Authentication* chapter for these configuration details.

See *Running Configuration* section for examples.

To configure MAB, use the **mab** command in dot1x profile configuration sub mode.

Configuration Example for MAB

Enable MAB:

```
Router#configure
Router(config)#dot1x profile test_mab
Router(dot1xx-test_mab)#mab
Router(dot1xx-test_mab)#commit
```

Configure the authenticator retry time for MAB clients:

```
Router#configure
Router(config)#dot1x profile test_mab
Router(dot1xx-test_mab)#authenticator
Router(dot1xx-test_mab-auth)#timer mab-retry-time 60
Router(dot1xx-test_mab-auth)#commit
```

Attach the dot1x profile to the corresponding interface or port on the router.

```
Router(config)#interface GigabitEthernet0/0/0/0
Router(config-intf)#dot1x profile test_mab
Router(config-intf)#commit
```

Running Configuration

```
Router# show running-configuration

!
radius-server host <ip-address> auth-port <auth-port-num> acct-port <acct-port-num>
 key 7 <key>
!
aaa authentication dot1x default group radius
interface GigabitEthernet0/0/0/0
 dot1x profile test_mab
!

dot1x profile test_mab
 mab
 authenticator
  timer reauth-time 60
  timer mab-retry-time 60
  host-mode single-host
```

```
  server dead action auth-retry
 !
!
end
```

Verify MAB Configuration

You can use these **show** commands to verify your MAB configuration:

- To check the MAB summary:

```
Router#show mab summary
Fri Apr  1 16:37:32.340 IST

NODE: node0_0_CPU0
==============================================================================
   Interface-Name        Client          Status
==============================================================================
     Gi0/0/0/0         1122.3344.5566   Authorized
Router#
```

- To verify the detailed MAB status:

```
Router#show mab detail
Fri Apr  1 16:37:37.140 IST

NODE: node0_0_CPU0

MAB info for GigabitEthernet0/0/0/0
------------------------------------------------------------------
InterfaceName       : Gi0/0/0/0
InterfaceHandle     : 0x00000060
HostMode            : single-host
PortControl         : Enabled
PuntState           : Stop Success
PuntSummary         : Punt disabled
Client:
  MAC Address       : 1122.3344.5566
  Status            : Authorized
  SM State          : Terminate
  ReauthTimeout     : 60s, Remaining 0 day(s), 00:00:46
  RetryTimeout      : 60s, timer not started yet
  AuthMethod        : PAP (remote)
  LastAuthTime      : 2022 Apr 01 16:37:23.634
  ProgrammingStatus : Add Success
Router#
```

- To verify the MAB interface summary:

```
Router#show mab interface gigabitEthernet 0/0/0/0
Fri Apr  1 16:38:27.715 IST
==============================================================================
   Interface-Name        Client          Status
==============================================================================
     Gi0/0/0/0         1122.3344.5566   Authorized
```

- To verify the MAB interface details:

```
Router#show mab interface gigabitEthernet 0/0/0/0 detail
Fri Apr  1 16:38:31.543 IST
MAB info for GigabitEthernet0/0/0/0
```

```
------------------------------------------------------------
InterfaceName       : Gi0/0/0/0
InterfaceHandle     : 0x00000060
HostMode            : single-host
PortControl         : Enabled
PuntState           : Stop Success
PuntSummary         : Punt disabled
Client:
  MAC Address       : 1122.3344.5566
  Status            : Authorized
  SM State          : Terminate
  ReauthTimeout     : 60s, Remaining 0 day(s), 00:00:51
  RetryTimeout      : 60s, timer not started yet
  AuthMethod        : PAP (remote)
  LastAuthTime      : 2022 Apr 01 16:38:23.640
  ProgrammingStatus : Add Success
Router#
```

- To verify the MAB interface statistics:

```
Router#show mab statistics interface gigabitEthernet 0/0/0/0
Fri Apr  1 16:41:23.011 IST
InterfaceName         : GigabitEthernet0/0/0/0
------------------------------------------------------------
MAC Learning:
  RxTotal             : 0
  RxNoSrcMac          : 0
  RxNoIdb             : 0
Port Control:
  EnableSuccess       : 1
  EnableFail          : 0
  UpdateSuccess       : 0
  UpdateFail          : 0
  PuntStartSuccess    : 0
  PuntStartFail       : 0
  PuntStopSuccess     : 1
  PuntStopFail        : 0
  AddClientSuccess    : 1
  AddClientFail       : 0
  RemoveClientSuccess : 0
  RemoveClientFail    : 0
Client                :
  MAC Address         : 1122.3344.5566
  Authentication:
    Success           : 1406
    Fail              : 0
    Timeout           : 0
    AAA Unreachable   : 0
Router#
```

System Logs for MAB

The router displays the following system logs on the console in various MAB scenarios:

- When the dot1x profile is applied on the port, with MAB feature enabled:

  Success case:

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with
Single-Host mode
```

  Failure case:

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_FAILURE : Hu0/0/1/0 : Failed to enable port-control
```

- When the dot1x profile is removed from the interface:

  Success case:

  ```
  %L2-DOT1X-5-PORT_CONTROL_DISABLE_SUCCESS : Hu0/0/1/0 : Port Control Disabled
  ```

  Failure case:

  ```
  %L2-DOT1X-5-PORT_CONTROL_DISABLE_FAILURE : Hu0/0/1/0 : Failed to disable port-control
  ```

- As part of MAB client authentication process:

  Success case:

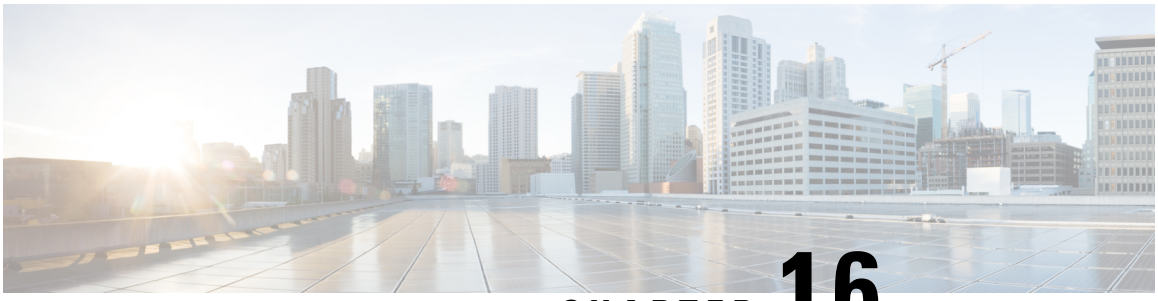
  ```
  %L2-DOT1X-5-MAB_AUTH_SUCCESS : Hu0/0/1/0 : Authentication successful for client
  <mac-address>
  %L2-DOT1X-5-PORT_CONTROL_ADD_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Enabled For Client
   <mac-address>
  ```

  Failure case:

  ```
  %L2-DOT1X-5-MAB_AUTH_FAIL    : Hu0/0/1/0 : Authentication failed for client <mac-address>
  %L2-DOT1X-5-PORT_CONTROL_REMOVE_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Disabled For
  Client <mac-address>
  ```

- When the authentication server is unreachable:

  ```
  %L2-DOT1X-5-MAB_AAA_UNREACHABLE : Hu0/0/1/0 : AAA server unreachable for client
  027E.15F2.CAE7, Retrying Authentication
  ```



CHAPTER 16

Cisco MASA Service

Table 49: Feature History Table

Feature Name	Release Information	Feature Description
Cisco MASA Service	IOS XR 7.8.1	The Cisco Manufacturer Authorized Signing Authority (MASA) service creates ownership vouchers (OVs) for a Cisco IOS XR router. These OVs along with the owner certificate (OC) **certify** that the router belongs to a given customer. Use cases where OVs and OCs are required include secure ZTP workflows and securely booting up your device on a 5G cell site over a third-party ethernet service. You can use the MASA service to download, and view logging and audit of OVs for the routers you own. This service also enables Cisco's Account teams to assign the serial number of a device to customers and view details of the logging, verification, and audit of OVs.

Key Terms and Concepts

Authentication Flow: The purpose of the Authentication flow is to identify and authenticate the router when it boots up. During this flow, the router also checks if the network can be trusted. The router does this by:

- validating the OV it received during the bootstrapping process and
- verifying the signature on the onboarding information with the owner certificate it received during the bootstrapping process.

The workflow involves the router booting to dynamically obtain the OV from MASA via the customer's staging or management servers

MASA Service: There are many services that require the ownership of the router to be authenticated, so it can be trusted by the network. MASA is a service run by Cisco to create and log OVs that are then used to validate the ownership of the router.

Owner Certificate: The OC is an X.509 certificate [RFC5280] that is used to identify an *owner*, for example, an organization. The OC can be signed by any certificate authority (CA).

The OC is used by a router to verify the CA signature using the public key that is also in the owner certificate.

The OC structure must contain the owner certificate itself, as well as all intermediate certificates leading to the "pinned-domain-cert" (PDC) certificate specified in the ownership voucher.

Ownership Voucher: The ownership voucher (OV) [RFC8366] is used to securely identify the router's owner, as known to the manufacturer. The ownership voucher is signed by the device's manufacturer.

The OV is used to verify that the owner certificate has a chain of trust leading to the trusted certificate (PDC) included in the ownership voucher.

pinned-domain-cert: The PDC field present in the OV typically pins a domain certificate, such as the certificate of a domain CA.

- Why Do I Need Cisco MASA?, on page 322
- Use Cases for Ownership Vouchers, on page 322
- Authentication Flow, on page 323
- Interacting with the MASA Server, on page 325
- Workflow to Provision a Router Using Ownership Voucher, on page 331

Why Do I Need Cisco MASA?

The Cisco MASA service securely authorizes ownership of a router so that the router can then establish a secure connection to the router owner's (your) network infrastructure.

The establishment of the ownership of the router is achieved through an Authentication Flow that on successful completion generates an ownership voucher (OV). The primary purpose of the OV is to securely convey a certificate—the "pinned-domain-cert" (PDC), that the router can then use to authenticate subsequent interactions with the network, for example, secure bootstrapping. Establishing ownership is important to the bootstrapping mechanisms so that the router can authenticate the network that is trying to take control of it.

Use Cases for Ownership Vouchers

The following use cases show examples where ownership vouchers apply:

- **Secure Zero Touch Provisioning (ZTP) Bootstrapping**

  Secure ZTP requires the ability to securely bootstrap a router over an untrusted network. This requires the ability of MASA to provide an OV to the router. The OV is used to authenticate the router to ensure connectivity of the router to the network.

  For more information on Secure ZTP, see the Secure Zero Touch Provisioning chapter in the *System Setup and Software Installation Guide for NCS 540 Series Routers*.

Note MASA can help generate OVs for Cisco Routers only.

- **Application Hosting on XR**

  Cisco IOS XR's Application Hosting (App Hosting) capability provides an IOS XR container on the router. This allows an application that augments XR features to be deployed. These applications can fall in one of the following categories:

  - Customer Apps—developed by Cisco's customers and cannot be signed by Cisco.
  - Partner Apps—developed by partners and are signed by Cisco.
  - Cisco App—developed by Cisco and signed by Cisco.

  You can use MASA in conjunction with the Golden ISO Tool (gisobuild.py) to provide the OVs to enable secure workflows for onboarding third party RPMs on router running Cisco IOS XR.

  For more information, see the *Application Hosting Guide for Cisco 8000 Series Routers*.

- **Deploy Router Using BootZ**

  Bootz is a secure zero-touch provisioning solution for data centers that automates the setup of network devices while ensuring robust security. It enables devices to connect and authenticate with the Bootz server, safeguarding the onboarding process against unauthorized access and cyber threats, streamlining remote device configuration without compromising safety.

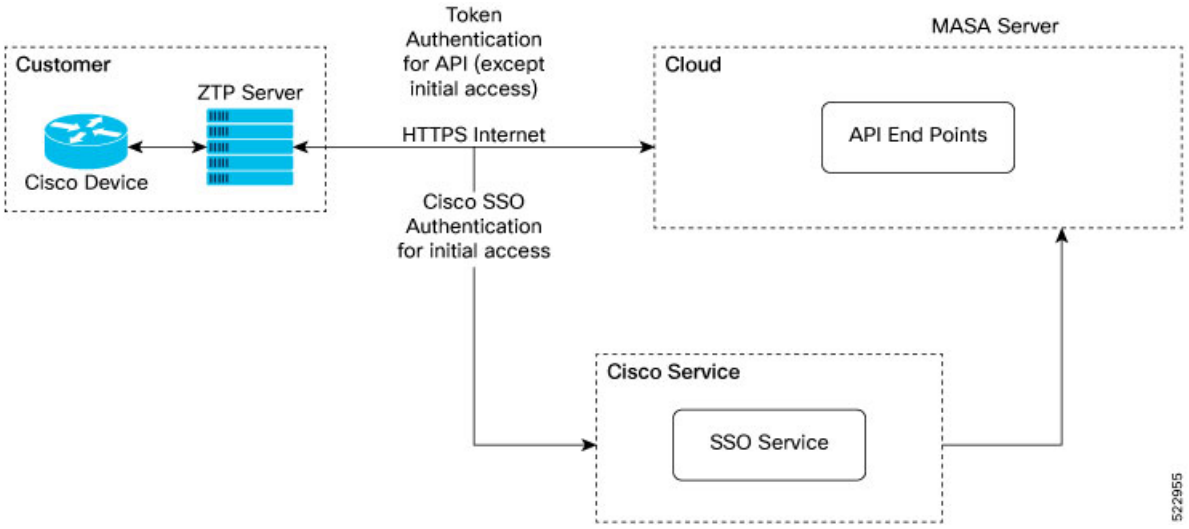
  Bootz uses a MASA to issue OVs that authenticate network devices during zero-touch provisioning.

  For more information on BootZ, see the Deploying Router Using Bootz Protocol chapter in the *System Setup and Software Installation Guide for Cisco 8000 Series Routers*.

Authentication Flow

The following figure is a high-level overview of different components involved in the authentication flow.

Figure 18: Components of the Authentication Flow



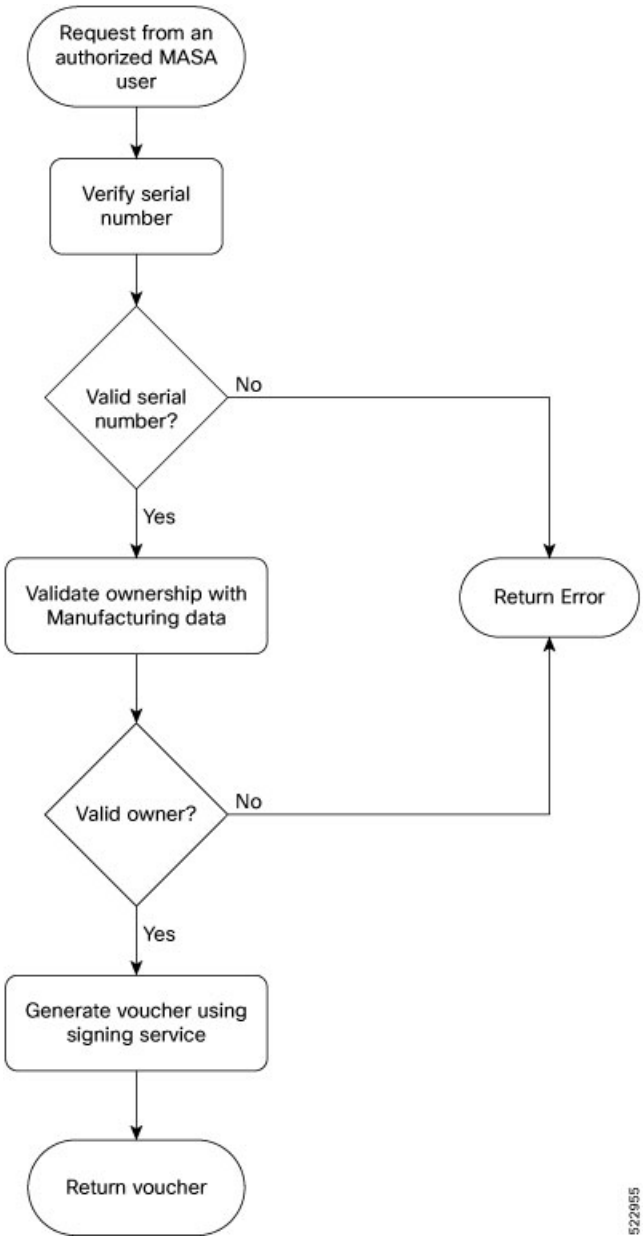
You can interact with the MASA Server web application through the ZTP Server to request, manage, and download the OVs for your routers.

The Zero Touch Provisioning (ZTP) server is to used to make a REST API call to the MASA Server.

The MASA Server authenticates the user, and on successful validation, generates the OVs.

The following figure illustrates the typical workflow to obtain the OVs.

Figure 19: Workflow to Obtain Ownership Vouchers



Interacting with the MASA Server

There are two ways to interact with the MASA server:

- through Web Application
- through REST API calls

Entities

The following entities interact with the MASA Server:

- **Organization**—A group in MASA specific to a Cisco customer. Data and access for each Organization is available to members of that group only.
- **Admin**—One or more initially-designated member(s) of an Organization who can invite other members into that organization in MASA, set access restrictions, and adjust other organization level settings.
- **User**—Any non-admin member of an organization who can interact with MASA. A user must be invited into an organization by the Admin
  - By default, new users have view-only access.
  - The Admin assigns permissions to request, download, or archive ownership vouchers

Prerequisites for Interacting with MASA Server

1. You must be an authorized MASA User
   - You must have a Cisco account and an active invitation to access MASA for the first time.

Note Contact the Cisco Technical Assistance team or your Account team to get a Cisco account.

   - Initial authentication requires *Cisco Single Sign On* to the MASA web application (masa.cisco.com).

     For subsequent authentication, you can generate access keys called *tokens*. Token serve as an alternative authentication mechanism that can be passed along in the header of API calls.

Note To generate access keys for the first time, on masa.cisco.com, go to **Settings**→ **Tokens**. For subsequent sessions, use API calls to manage existing tokens or create new ones as long as an unexpired token is still available.

     The following is an example of using a token in a header of a REST API call.

```
‘Authorization: Bearer
637c98ddcc58c75f679a94d7f244777be05c6600923c4549bc5669b26e04f2bc
gAAAAABjfRr9hqndFqbuqes9OvcfgucApgxpmm9qoVmUidYEs-_Aziu7yue-10dazZ3Rxk6vJHYD2Je7Z-IOD1Zc7kYSuBTX0
6GcQvF2e3nSM-_F9BoltjxAHcXkoMgbqS4APFGi16LiWRyP2b1_0rZO-EaTKFLEldTLfMAmovPDkZZ5vbBwRS058PZN1vB3IZIZ
jftYYYi9H_grazfwnAImjKbQC6tjQw==
```

     Tokens can have a custom validity period of up to six months that can be revoked at any time. The scope of the tokens is limited to scope of your role.

2. ZTP server must be able to access the Internet

Note MASA application is served through HTTPS to provide a secure connection between the end user and the service.

User Permissions

The MASA Server supports Role Based Access Control and provides the following access:

- Regular user—By default, regular users have only read access to their organization. Admin users can provide additional privileges as required.
- Admin—Admin users have the ability to view and manage OVs for all routers in the database in their organization as well as other privileges as mentioned in the table below.

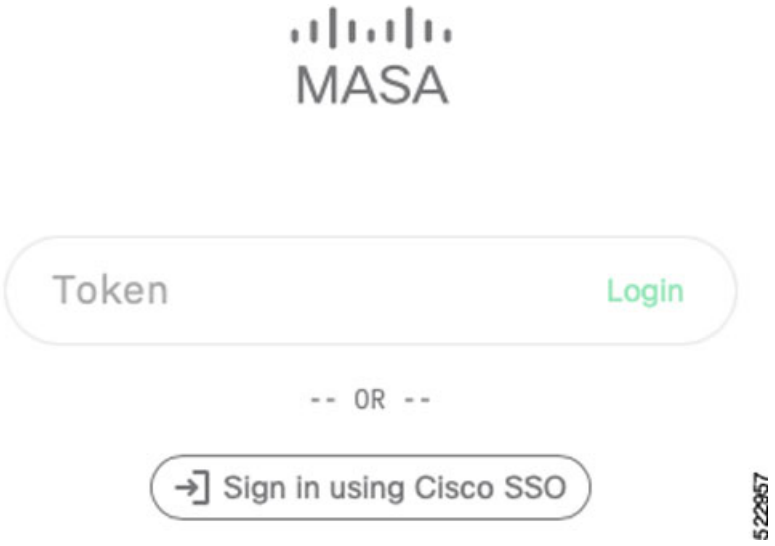
Table 50: User Permissions

Type	Regular User	Admin
Invite other People into the organization	Not allowed	Allowed by default
Add or remove permissions for other users	Not allowed	Allowed by default
View all existing vouchers	Allowed by default	Allowed by default
Request new vouchers	Permission can be provided by Admin	Allowed by default
Download vouchers	Permission can be provided by Admin	Allowed by default
Archive vouchers	Permission can be provided by Admin	Allowed by default

Interacting with MASA Through Web Application

1. Go to masa.cisco.com

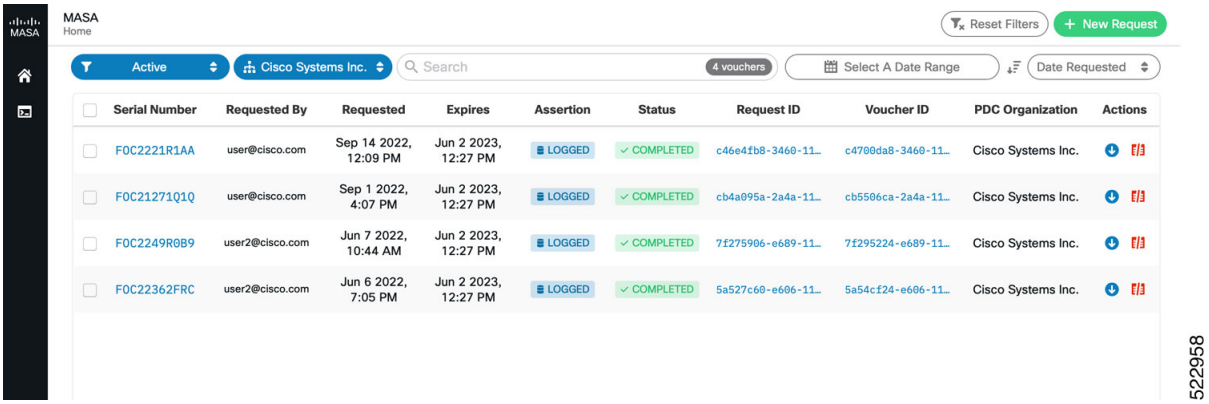
Figure 20: Sign in Page—MASA Web Application



2. Click **Sign in using Cisco** SSO.
3. Enter your username and password to access the application
4. Accept the End User License Agreement.

   The MASA Home page displays the status of any recent requests that were initiated and quick links to download any recently generated ownership vouchers.

Figure 21: Home Page—MASA Web Application



Requesting OVs for Your Router

1. Click **New Request** on the top right of the Home page.
2. In the New Request dialog box, enter details for one of the following:
   - Serial number of your router

     You can get the serial number from the bottom of your router; it is an 11 digit alphanumeric string. You can also get the serial number by running the **show version** command on your router.

- Pinned-domain Certificate

  There are multiple ways to generate a PDC (.pem). For example, through OpenSSL. You can either paste the content of the certificate directly or browse to a file that contains the PDC.

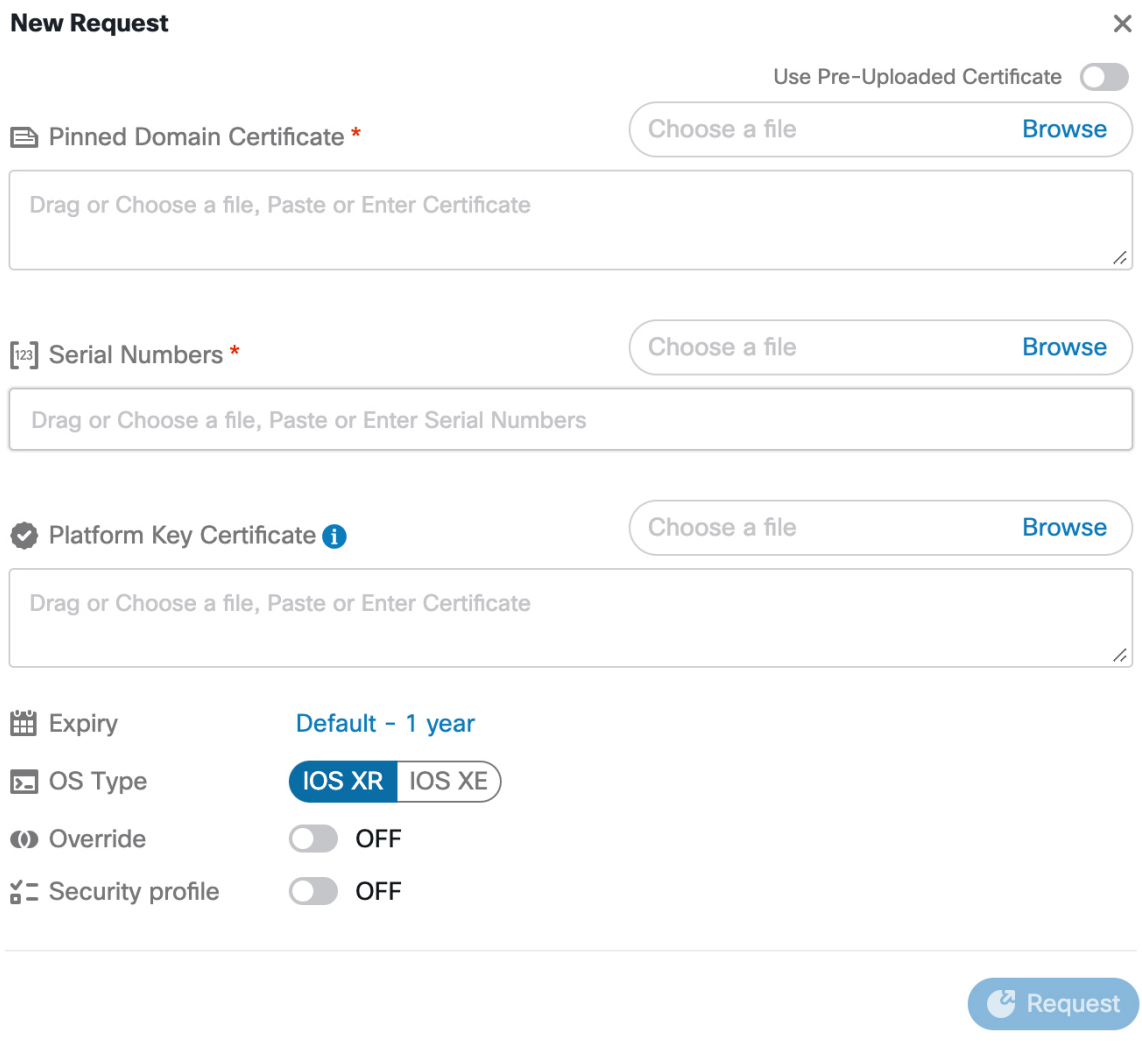
  You can pre-upload the certificate prior to requesting the OV.

  To select the pre-uploaded certificate while requesting OV, turn on the toggle button named *use pre-uploaded certificate*.You can see the already uploaded certificates here, you can select the certificate from this list.

- Serial number of one or more routers for which you want the OVs.

Note Always use the serial number of the route processor (RP) of your router.

Figure 22: New Request Page



523805

Figure 23: Home Page—With New OVs Displayed

522960

Depending on your user permissions, you can perform the following actions from the Home page.

- Download the generated OVs.
- Regenerate OVs.
- View details of past requests
- Filter, sort, and group the requests based on their attributes
- Archive the OVs.

Interacting with MASA Through REST APIs

You can also use APIs to programmatically interact with the MASA service.

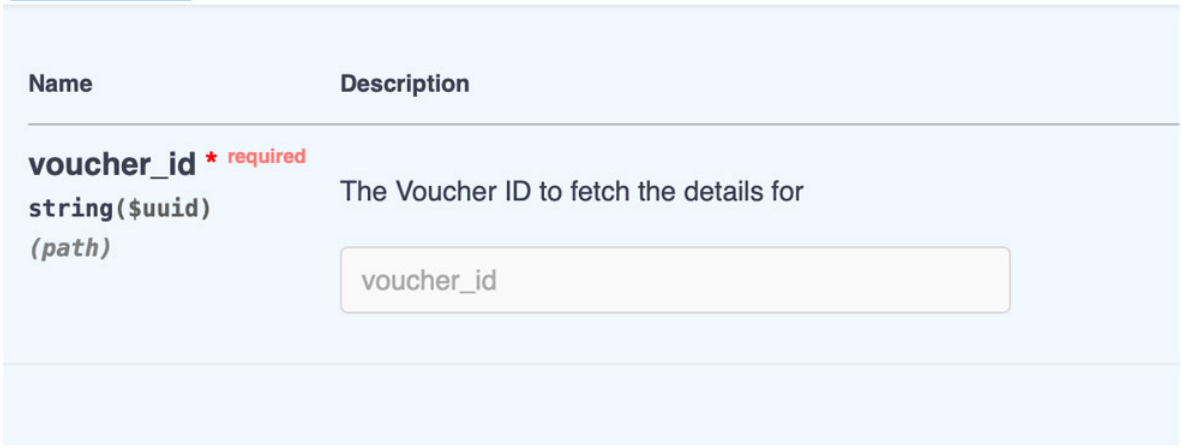
See the OpenAPI documentation page that contains details about the paths, formats, and structures of the APIs.

For example, use this API to request for the ownership voucher:

```
POST /request/ov
```

Use this API to fetch details about an already generated voucher:

```
GET /voucher/{voucher_id}
```

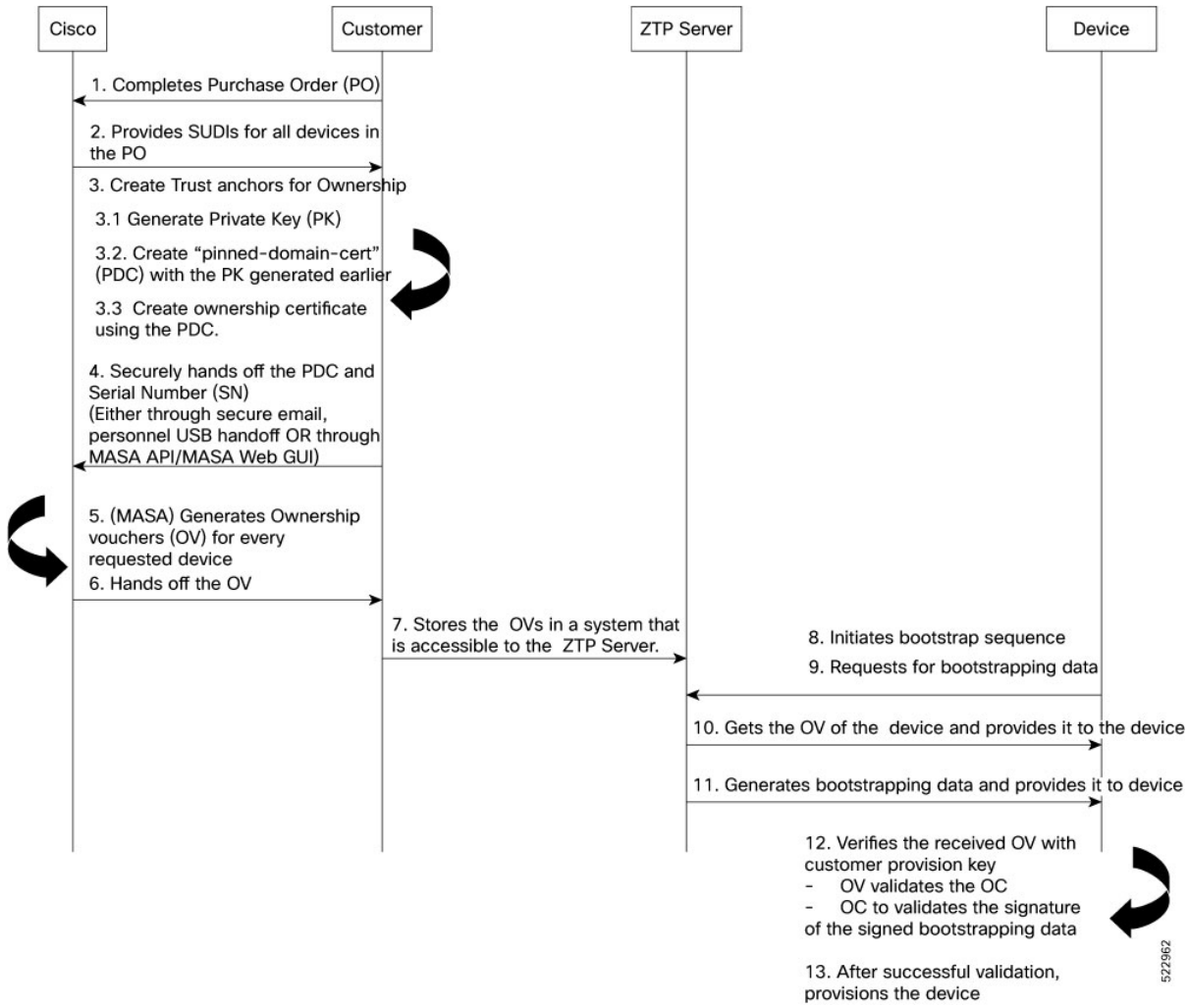



```
Response:
{
  "ok": true,
  "voucher": {
    "req_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "voucher_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "requested_at": "2022-08-31T09:43:39.719Z",
    "created_at": "2022-08-31T09:43:39.719Z",
    "expires_at": "2022-08-31T09:43:39.719Z",
    "last_renewal_at": "2022-08-31T09:43:39.719Z",
    "assertion": "logged",
    "status": "completed",
    "serial_number": "T8I52J1IKOM",
    "pdc_organization": "Cisco Systems",
    "requested_by": "user1@cisco.com"
  }
}
```

Note “serial Number” is serial number of the route processor. You can provide up to 20 serial numbers in a single request.

Workflow to Provision a Router Using Ownership Voucher

The following figure illustrates the complete workflow to provision a Cisco IOS XR router by using the ownership vouchers.


Cisco
Customer
ZTP Server
Device
1. Completes Purchase Order (PO)
2. Provides SUDIs for all devices in the PO
3. Create Trust anchors for Ownership
3.1 Generate Private Key (PK)
3.2. Create "pinned-domain-cert" (PDC) with the PK generated earlier
3.3 Create ownership certificate using the PDC.
4. Securely hands off the PDC and Serial Number (SN)
(Either through secure email, personnel USB handoff OR through MASA API/MASA Web GUI)
5. (MASA) Generates Ownership vouchers (OV) for every requested device
6. Hands off the OV
7. Stores the OVs in a system that is accessible to the ZTP Server.
8. Initiates bootstrap sequence
9. Requests for bootstrapping data
10. Gets the OV of the device and provides it to the device
11. Generates bootstrapping data and provides it to device
12. Verifies the received OV with customer provision key
- OV validates the OC
- OC to validates the signature of the signed bootstrapping data
13. After successful validation, provisions the device
522962