Cisco IOS XR System Security Configuration Guide

Cisco IOS XR Software Release 3.5

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

Text Part Number: OL-12287-01

Cisco IOS XR System Security Configuration Guide

CONTENTS

Preface xi

Changes to This Document xi

Obtaining Documentation, Obtaining Support, and Security Guidelines xi

Implementing Certification Authority Interoperability on Cisco IOS XR Software SC-1

Contents SC-1

Prerequisites for Implementing Certification Authority SC-2

Restrictions for Implementing Certification Authority SC-2

Information About Implementing Certification Authority SC-2

Supported Standards for Certification Authority Interoperability SC-2

Certification Authorities SC-3

How to Implement CA Interoperability SC-5

Configuring a Router Hostname and IP Domain Name SC-6

Generating an RSA Key Pair SC-7

Declaring a Certification Authority and Configuring a Trusted Point SC-8

Authenticating the CA SC-10

Requesting Your Own Certificates SC-11

Configuring Certificate Enrollment Using Cut-and-Paste SC-12

Configuration Examples for Implementing Certification Authority Interoperability SC-14

Configuring Certification Authority Interoperability: Example SC-14

Where to Go Next SC-16

Additional References SC-16

Related Documents SC-16

Standards SC-16

MIBs SC-17

RFCs SC-17

Technical Assistance SC-17

Implementing Internet Key Exchange Security Protocol on Cisco IOS XR Software SC-19

Contents SC-20

Prerequisites SC-20

Information About Implementing IKE Security Protocol Configurations for IPSec Networks SC-20

Supported Standards SC-21

Concessions for Not Enabling IKE SC-22

IKE Policies SC-22

ISAKMP Identity SC-26
ISAKMP Profile Overview SC-26
Mask Preshared Keys SC-27
Preshared Keys Using a AAA Server SC-27
Internet Key Exchange Mode Configuration SC-28
Banner, Auto-Update, and Browser-Proxy SC-29
Pushing a Configuration URL Through a Mode-Configuration Exchange SC-29
Internet Key Exchange Extended Authentication SC-30
Call Admission Control SC-30
Information About IP Security VPN Monitoring SC-31
Information About IKE for the Cisco IPSec VPN SPA on Cisco IOS XR Software SC-32
IPSec Dead Peer Detection Periodic Message Option SC-32
How to Implement IKE Security Protocol Configurations for IPSec Networks SC-32
Enabling or Disabling IKE SC-33
Configuring IKE Policies SC-34
Defining Group Policy Information for Mode Configuration SC-36
Configuring a Banner SC-40
Configuring Auto-Upgrade SC-40
Configuring a Browser Proxy SC-41
Configuring a Browser-Proxy Map to a Group SC-42
Configuring the Pushing of a Configuration URL Through a Mode-Configuration Exchange SC-43
Manually Configuring RSA Keys SC-44
Configuring ISAKMP Preshared Keys in ISAKMP Keyrings SC-48
Configuring Call Admission Control SC-50
Configuring Crypto Keyrings SC-54
Configuring IP Security VPN Monitoring SC-57
How to Implement IKE for Locally Sourced and Destined Traffic SC-58
Configuring the ISAKMP Profile for Locally Sourced and Destined Traffic SC-58
How to Implement IKE for Cisco IPSec VPN SPAs on Cisco IOS XR Software SC-62
Configuring a Periodic Dead Peer Detection Message SC-63
Configuring the ISAKMP Profile for Service Interfaces SC-64
Configuration Examples for Implementing IKE Security Protocol SC-68
Creating IKE Policies: Example SC-69
Configuring a service-ipsec Interface with a Dynamic Profile: Example SC-69
Configuring Easy VPN with a Local AAA: Example SC-70
Configuring VRF-Aware: Example SC-71
Additional References SC-73
Related Documents SC-73
Standards SC-73

MIBs SC-74
RFCs SC-74
Technical Assistance SC-74

Implementing Keychain Management on Cisco IOS XR Software SC-75

Contents SC-75
Restrictions for Implementing Keychain Management SC-75
Information About Implementing Keychain Management SC-76
Lifetime of a Key SC-76
How to Implement Keychain Management SC-76
Configuring a Keychain SC-77
Configuring a Tolerance Specification to Accept Keys SC-78
Configuring a Key Identifier for the Keychain SC-79
Configuring the Text for the Key String SC-81
Determining the Valid Keys SC-82
Configuring the Keys to Generate Authentication Digest for the Outbound Application Traffic SC-84
Configuring the Cryptographic Algorithm SC-85
Configuration Examples for Implementing Keychain Management SC-87
Configuring Keychain Management: Example SC-87
Additional References SC-88
Related Documents SC-88
Standards SC-88
MIBs SC-89
RFCs SC-89
Technical Assistance SC-89

Implementing IPSec Network Security on Cisco IOS XR Software SC-91

Contents SC-92
Prerequisites for Implementing IPSec Network Security SC-92
Restrictions for Implementing IPSec Network Security SC-93
Restrictions for Implementing IPSec Network with a Cisco IPSec VPN SPA SC-93
Information About Implementing IPSec Networks SC-94
Crypto Profiles SC-94
Dynamic Crypto Profiles SC-95
Crypto Access Lists SC-95
Transform Sets SC-96
Global Lifetimes for IPSec Security Associations SC-96
Manual IPSec Security Associations SC-97

Perfect Forward Secrecy SC-97
Checkpointing SC-98
DF Bit Override Functionality with IPSec Tunnels SC-98
IPSec Antireplay Window SC-98
IPSec NAT Transparency SC-99
IPSec Security Association Idle Timers SC-99
Prefragmentation for Cisco IPSec VPN SPAs SC-99
Reverse-Route Injection SC-100
IPSec—SNMP Support SC-101
Information About an IPSec Network with a Cisco IPSec VPN SPA on Cisco IOS XR Software SC-101
Cisco IPSec VPN SPA Overview SC-101
Displaying the SPA Hardware Type SC-101
Information About Security for VPNs with IPSec SC-102
How to Implement General IPSec Configurations for IPSec Networks SC-104
Setting Global Lifetimes for IPSec Security Associations SC-105
Creating Crypto Access Lists SC-106
Defining Transform Sets SC-108
Configuring Crypto Profiles SC-109
Configuring the DF Bit for the Encapsulating Header in IPSec Tunnels SC-114
Configuring the IPSec Antireplay Window: Expanding and Disabling SC-115
Configuring IPSec NAT Transparency SC-118
Configuring IPSec Security Association Idle Timers SC-120
Disabling Prefragmentation for Cisco IPSec VPN SPAs SC-124
Configuring Reverse-Route Injection in a Crypto Profile SC-127
Configuring IPSec Failure History Table Size SC-128
How to Implement IPSec Network Security for Locally Sourced and Destined Traffic SC-129
Applying Crypto Profiles to tunnel-ipsec Interfaces SC-130
Applying Crypto Profiles to Crypto Transport SC-131
How to Implement IPSec Network Security for VPNs SC-132
Configuring IPSec Virtual Interfaces SC-133
Configuring the Default Path Maximum Transmission Unit for the SA SC-139
Configuration Examples for Implementing IPSec Network Security for Locally Sourced Traffic and Destined Traffic SC-140
Configuring a Static Profile and Attaching to a Tunnel-ipsec Interface: Example SC-140
Configuring a Dynamic Profile and Attaching to a Tunnel-ipsec Interface: Example SC-141
Configuring a Static Profile and Attaching to Transport: Example SC-142
Configuration Examples for an IPSec Network with a Cisco IPSec VPN SPA SC-142
Configuring IPSec for a VRF-aware Service-ipsec Interface: Example SC-142

Configuring a Service-gre Interface: Example SC-145
Additional References SC-147
Related Documents SC-147
Standards SC-147
MIBs SC-147
RFCs SC-148
Technical Assistance SC-148

Implementing Secure Shell on Cisco IOS XR Software SC-149
Contents SC-149
Prerequisites to Implementing Secure Shell SC-150
Restrictions for Implementing Secure Shell SC-150
Information About Implementing Secure Shell SC-151
SSH Server SC-151
SSH Client SC-151
SFTP Feature Overview SC-151
AAA Feature SC-152
How to Implement Secure Shell SC-152
Configuring SSH SC-152
Configuring the SSH Client SC-154
Configuration Examples for Implementing Secure Shell SC-156
Configuring Secure Shell: Example SC-156
Additional References SC-156
Related Documents SC-156
Standards SC-157
MIBs SC-157
RFCs SC-157
Technical Assistance SC-158

Implementing Secure Socket Layer on Cisco IOS XR Software SC-159
Contents SC-160
Prerequisites for Implementing Secure Socket Layer SC-160
Information About Implementing Secure Socket Layer SC-160
Purpose of Certification Authorities SC-160
How to Implement Secure Socket Layer SC-161
Configuring Secure Socket Layer SC-161
Configuration Examples for Implementing Secure Socket Layer SC-164
Configuring Secure Socket Layer: Example SC-164
Additional References SC-164

Related Documents SC-164
Standards SC-165
MIBs SC-165
RFCs SC-165
Technical Assistance SC-165

Configuring AAA Services on Cisco IOS XR Software SC-167

Contents SC-168
Prerequisites for Configuring AAA Services SC-169
Restrictions for Configuring AAA Services SC-169
Information About Configuring AAA Services SC-169
User, User Groups, and Task Groups SC-170
Cisco IOS XR Software Administrative Model SC-172
Password Types SC-177
Task-Based Authorization SC-178
Task IDs for TACACS+ and RADIUS Authenticated Users SC-179
XML Schema for AAA Services SC-181
About RADIUS SC-182
How to Configure AAA Services SC-183
Configuring Task Groups SC-184
Configuring User Groups SC-186
Configuring Users SC-188
Configuring Router to RADIUS Server Communication SC-190
Configuring RADIUS Dead-Server Detection SC-194
Configuring Per VRF AAA SC-196
Configuring a TACACS+ Server SC-198
Configuring RADIUS Server Groups SC-201
Configuring TACACS+ Server Groups SC-203
Configuring AAA Method Lists SC-204
Applying Method Lists for Applications SC-216
Configuring Login Parameters SC-220
Configuration Examples for Configuring AAA Services SC-221
Configuring AAA Services: Example SC-221
Additional References SC-223
Related Documents SC-223
Standards SC-223
MIBs SC-223
RFCs SC-224
Technical Assistance SC-224

Configuring Software Authentication Manager on Cisco IOS XR Software SC-225

Implementing Management Plane Protection on Cisco IOS XR Software SC-227

Contents SC-227

Restrictions for Implementing Management Plane Protection SC-228

Information About Implementing Management Plane Protection SC-228

Inband Management Interface SC-228

Control Plane Protection Overview SC-228

Management Plane SC-228

Management Plane Protection Feature SC-229

Benefits of the Management Plane Protection Feature SC-229

How to Configure a Device for Management Plane Protection SC-229

Configuring a Device for Management Plane Protection SC-230

Configuration Examples for Implementing Management Plane Protection SC-232

Configuring Management Plane Protection: Example SC-232

Additional References SC-233

Related Documents SC-233

Standards SC-233

MIBs SC-233

RFCs SC-234

Technical Assistance SC-234

Index

Preface

This guide describes the configuration and examples for system security.

For system security command descriptions, usage guidelines, task IDs, and examples, refer to the *Cisco IOS XR System Security Command Reference*.

The preface contains the following sections:

- Changes to This Document, page xi
- Obtaining Documentation, Obtaining Support, and Security Guidelines, page xi

Changes to This Document

Table 1 lists the technical changes made to this document since it was first printed.

Table 1 Changes to This Document

Revision	Date	Change Summary
OL-12287-01	June 2007	Initial release of this document.

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Implementing Certification Authority Interoperability on Cisco IOS XR Software

Certification authority (CA) interoperability is provided in support of the IP Security (IPSec), Secure Socket Layer (SSL), and Secure Shell (SSH) protocols. CA interoperability permits Cisco IOS XR devices and CAs to communicate so that your Cisco IOS XR 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.

This module describes the tasks that you need to implement CA interoperability on your Cisco IOS XR network.

Note For a complete description of the public key infrastructure (PKI) commands used in this chapter, refer to the *Public Key Infrastructure Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference*. To locate documentation for other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing Certification Authority Interoperability on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.3.0	No modification.
Release 3.4.0	A procedure was added on how to declare the trustpoint certification authority (CA) for both the Cisco CRS-1 and Cisco XR 12000 Series Router.
Release 3.5.0	No modification.

Contents

- Prerequisites for Implementing Certification Authority, page SC-2
- Restrictions for Implementing Certification Authority, page SC-2
- Information About Implementing Certification Authority, page SC-2
- How to Implement CA Interoperability, page SC-5

- Configuration Examples for Implementing Certification Authority Interoperability, page SC-14
- Additional References, page SC-16

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 for security commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- You must install and activate the Package Installation Envelope (PIE) for the security software.

  For detailed information about optional PIE installation, refer to the *Cisco IOS XR System Management Guide*.
- You need to have a CA available to your network before you configure this interoperability feature. The CA must support Cisco Systems PKI protocol, the Simple Certificate Enrollment Protocol (SCEP) (formerly called certificate enrollment protocol [CEP]).

Restrictions for Implementing Certification Authority

Cisco IOS XR software does not support CA server public keys greater than 2048 bits.

Information About Implementing Certification Authority

To implement CA, you need to understand the following concepts:

- Supported Standards for Certification Authority Interoperability, page SC-2
- Certification Authorities, page SC-3

Supported Standards for Certification Authority Interoperability

Cisco supports the following standards:

- IPSec—IP Security Protocol. IPSec is a framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPSec provides these security services at the IP layer; it uses Internet Key Exchange (IKE) to handle negotiation of protocols and algorithms based on local policy, and to generate the encryption and authentication keys to be used by IPSec. IPSec can be used to protect one or more data flows between a pair of hosts, a pair of security gateways, or a security gateway and a host.
- IKE—A hybrid protocol that implements Oakley and Skeme key exchanges inside the Internet Security Association Key Management Protocol (ISAKMP) framework. Although IKE can be used with other protocols, its initial implementation is with the IPSec protocol. IKE provides authentication of the IPSec peers, negotiates IPSec keys, and negotiates IPSec security associations (SAs).
- Public-Key Cryptography Standard #7 (PKCS #7)—A standard from RSA Data Security Inc. used to encrypt and sign certificate enrollment messages.

- Public-Key Cryptography Standard #10 (PKCS #10)—A standard syntax from RSA Data Security Inc. for certificate requests.
- RSA keys—RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Adelman. RSA keys come in pairs: one public key and one private key.
- SSL—Secure Socket Layer protocol.
- X.509v3 certificates—Certificate support that allows the IPSec-protected network to scale by providing the equivalent of a digital ID card to each device. When two devices want to communicate, they exchange digital certificates to prove their identity (thus removing the need to manually exchange public keys with each peer or specify a shared key at each peer). These certificates are obtained from a CA. X.509 is part of the X.500 standard of the ITU.

Certification Authorities

The following sections provide background information about CAs:

- Purpose of CAs, page SC-3
- IPSec Without CAs, page SC-4
- IPSec with CAs, page SC-4
- IPSec with Multiple Trustpoint CAs, page SC-4
- How CA Certificates Are Used by IPSec Devices, page SC-5
- CA Registration Authorities, page SC-5

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.

IPSec Without CAs

Without a CA, if you want to enable IPSec services (such as encryption) between two Cisco routers, you must first ensure that each router has the key of the other router (such as an RSA public key or a shared key). This requirement means that you must manually perform one of the following operations:

- At each router, enter the RSA public key of the other router.
- At each router, specify a shared key to be used by both routers.

If you have multiple Cisco routers in a mesh topology and want to exchange IPSec traffic passing among all of those routers, you must first configure shared keys or RSA public keys among all of those routers.

Every time a new router is added to the IPSec network, you must configure keys between the new router and each of the existing routers.

Consequently, the more devices there are that require IPSec services, the more involved the key administration becomes. This approach does not scale well for larger, more complex encrypting networks.

IPSec with CAs

With a CA, you need not configure keys between all the encrypting routers. Instead, you individually enroll each participating router with the CA, requesting a certificate for the router. When this enrollment has been accomplished, each participating router can dynamically authenticate all the other participating routers.

To add a new IPSec router to the network, you need only configure that new router to request a certificate from the CA, instead of making multiple key configurations with all the other existing IPSec routers.

IPSec with Multiple Trustpoint CAs

With multiple trustpoint CAs, you no longer have to enroll a router with the CA that issued a certificate to a peer. Instead, you configure a router with multiple CAs that it trusts. Thus, a router can use a configured CA (a trusted root) to verify certificates offered by a peer that were not issued by the same CA defined in the identity of the router.

Configuring multiple CAs allows two or more routers enrolled under different domains (different CAs) to verify the identity of each other when using IKE to set up IPSec tunnels.

Through SCEP, each router is configured with a CA (the enrollment CA). The CA issues a certificate to the router that is signed with the private key of the CA. To verify the certificates of peers in the same domain, the router is also configured with the root certificate of the enrollment CA.

To verify the certificate of a peer from a different domain, the root certificate of the enrollment CA in the domain of the peer must be configured securely in the router.

During IKE phase one signature verification, the initiator will send the responder a list of its CA certificates. The responder should send the certificate issued by one of the CAs in the list. If the certificate is verified, the router saves the public key contained in the certificate on its public key ring.

With multiple root CAs, Virtual Private Network (VPN) users can establish trust in one domain and easily and securely distribute it to other domains. Thus, the required private communication channel between entities authenticated under different domains can occur.

How CA Certificates Are Used by IPSec Devices

When two IPSec routers want to exchange IPSec-protected traffic passing between them, they must first authenticate each other—otherwise, IPSec protection cannot occur. The authentication is done with IKE.

Without a CA, a router authenticates itself to the remote router using either RSA-encrypted nonces or preshared keys. Both methods require keys to have been previously configured between the two routers.

With a CA, a router authenticates itself to the remote router by sending a certificate to the remote router and performing some public key cryptography. Each router must send its own unique certificate that was issued and validated by the CA. This process works because the certificate of each router encapsulates the public key of the router, each certificate is authenticated by the CA, and all participating routers recognize the CA as an authenticating authority. This scheme is called IKE with an RSA signature.

Your router can continue sending its own certificate for multiple IPSec sessions and to multiple IPSec peers until the certificate expires. When its certificate expires, the router administrator must obtain a new one from the CA.

When your router receives a certificate from a peer from another domain (with a different CA), the certificate revocation list (CRL) downloaded from the CA of the router does not include certificate information about the peer. Therefore, you should check the CRL published by the configured trustpoint with the Lightweight Directory Access Protocol (LDAP) URL to ensure that the certificate of the peer has not been revoked.

To query the CRL published by the configured trustpoint with the LDAP URL, use the **query url** command in trustpoint configuration mode.

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.

How to Implement CA Interoperability

This section contains the following procedures:

- Configuring a Router Hostname and IP Domain Name, page SC-6 (required)
- Generating an RSA Key Pair, page SC-7 (required)
- Declaring a Certification Authority and Configuring a Trusted Point, page SC-8 (required)
- Authenticating the CA, page SC-10 (required)
- Requesting Your Own Certificates, page SC-11 (required)
- Configuring Certificate Enrollment Using Cut-and-Paste, page SC-12

Configuring a 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.

SUMMARY STEPS

1. **configure**
2. **hostname** *name*
3. **domain name** *domain-name*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enables global configuration mode.
Step 2	**hostname** *name* **Example:** `RP/0/RP0/CPU0:router(config)# hostname myhost`	Configures the hostname of the router.

	Command or Action	Purpose
Step 3	**domain name** ***domain-name*** **Example:** `RP/0/RP0/CPU0:router(config)# domain name mydomain.com`	Configures the IP domain name of the router.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Generating an RSA Key Pair

This task generates an RSA key pair.

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.

SUMMARY STEPS

1. **crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*]
2. **crypto key zeroize rsa** [*keypair-label*]
3. **show crypto key mypubkey rsa**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto key generate rsa** [**usage keys** | **general-keys**] [***keypair-label***] **Example:** `RP/0/RP0/CPU0:router# crypto key generate rsa general-keys`	Generates RSA key pairs. • Use the **usage keys** keyword to specify special usage keys; use the **general-keys** keyword to specify general-purpose RSA keys. • The *keypair-label* argument is the RSA key pair label that names the RSA key pairs.
Step 2	**crypto key zeroize rsa** [***keypair-label***] **Example:** `RP/0/RP0/CPU0:router# crypto key zeroize rsa key1`	(Optional) Deletes all RSAs from the router. • Under certain circumstances, you may want to delete all RSA keys from you router. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys. • To remove a specific RSA key pair, use the *keypair-label* argument.
Step 3	**show crypto key mypubkey rsa** **Example:** `RP/0/RP0/CPU0:router# show crypto key mypubkey rsa`	(Optional) Displays the RSA public keys for your router.

Declaring a Certification Authority and Configuring a Trusted Point

This task declares a CA and configures a trusted point.

SUMMARY STEPS

1. **configure**
2. **crypto ca trustpoint** *ca-name*
3. **enrollment url** *CA-URL*
4. **query url** *LDAP-URL*
5. **enrollment retry period** *minutes*
6. **enrollment retry count** *number*
7. **rsakeypair** *keypair-label*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ca trustpoint** ***ca-name*** **Example:** `RP/0/RP0/CPU0: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:** `RP/0/RP0/CPU0: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:** `RP/0/RP0/CPU0: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:** `RP/0/RP0/CPU0: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:** `RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 7	**rsakeypair** ***keypair-label*** **Example:** `RP/0/RP0/CPU0: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	**end**
	 or
	 commit
	
	 Example:
	 `RP/0/RP0/CPU0:router(config-trustp)# end`
	 or
	 `RP/0/RP0/CPU0:router(config-trustp)# commit`
		Saves configuration changes.
		 • When you issue the **end** command, the system prompts you to commit changes:
		 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		 – Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		 – Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		 – Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		 • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Authenticating the 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.

SUMMARY STEPS

1. **crypto ca authenticate** *ca-name*
2. **show crypto ca certificates**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto ca authenticate** ***ca-name*** **Example:** `RP/0/RP0/CPU0:router# crypto ca authenticate myca`	Authenticates the CA to your router by obtaining a CA certificate, which contains the public key for the CA.
Step 2	**show crypto ca certificates** **Example:** `RP/0/RP0/CPU0:router# show crypto ca certificates`	(Optional) Displays information about the CA certificate.

Requesting 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.

SUMMARY STEPS

1. **crypto ca enroll** *ca-name*
2. **show crypto ca certificates**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto ca enroll** ***ca-name*** **Example:** `RP/0/RP0/CPU0:router# crypto ca enroll myca`	Requests certificates for all of your RSA key pairs. • This command causes your router to request as many certificates as there are RSA key pairs, so you need only perform this command once, even if you have special usage RSA key pairs. • This command requires you to create a challenge password that is not saved with the configuration. This password is required if your certificate needs to be revoked, so you must remember this password. • A certificate may be issued immediately or the router sends a certificate request every minute until the enrollment retry period is reached and a timeout occurs. If a timeout occurs, contact your system administrator to get your request approved, and then enter this command again.
Step 2	**show crypto ca certificates** **Example:** `RP/0/RP0/CPU0:router# show crypto ca certificates`	(Optional) Displays information about the CA certificate.

Configuring 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.

SUMMARY STEPS

1. **configure**
2. **crypto ca trustpoint** *ca-name*
3. **enrollment terminal**
4. **end**
   or
   commit
5. **crypto ca authenticate** *ca-name*
6. **crypto ca enroll** *ca-name*
7. **crypto ca import** ca-*name* **certificate**
8. **show crypto ca certificates**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto ca trustpoint** *ca-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto ca trustpoint myca`
	`RP/0/RP0/CPU0:router(config-trustp)#`
		Declares the CA that your router should use and enters trustpoint configuration mode.
		• Use the *ca-name* argument to specify the name of the CA.

Step 3	**enrollment terminal** **Example:** `RP/0/RP0/CPU0:router(config-trustp)# enrollment terminal`	Specifies manual cut-and-paste certificate enrollment.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-trustp)# end`
	or
	`RP/0/RP0/CPU0:router(config-trustp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**crypto ca authenticate** *ca-name* **Example:** `RP/0/RP0/CPU0:router# crypto ca authenticate myca`	Authenticates the CA by obtaining the certificate of the CA. • Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2.

	Command or Action	Purpose
Step 6	**crypto ca enroll** *ca-name* **Example:** `RP/0/RP0/CPU0:router# crypto ca enroll myca`	Obtains the certificates for your router from the CA. • Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2.
Step 7	**crypto ca import** ca-*name* **certificate** **Example:** `RP/0/RP0/CPU0:router# crypto ca import myca certificate`	Imports a certificate manually at the terminal. • Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2. **Note** You must enter the **crypto ca import** command twice if usage keys (signature and encryption keys) are used. The first time the command is entered, one of the certificates is pasted into the router; the second time the command is entered, the other certificate is pasted into the router. (It does not matter which certificate is pasted first.
Step 8	**show crypto ca certificates** **Example:** `RP/0/RP0/CPU0:router# show crypto ca certificates`	Displays information about your certificate and the CA certificate.

Configuration Examples for Implementing Certification Authority Interoperability

This section provides the following configuration example:

- Configuring Certification Authority Interoperability: Example, page SC-14

Configuring Certification Authority Interoperability: Example

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
```

Where to Go Next

After you have finished configuring CA interoperability, you should configure IKE, IPSec, and SSL. IKE configuration is described in the *Implementing Internet Key Exchange Security Protocol on Cisco IOS XR Software* module, IPSec in the *Implementing IPSec Network Security on Cisco IOS XR Software* module, and SSL in the *Implementing Secure Socket Layer on Cisco IOS XR Software* module. These modules are located in *Cisco IOS XR System Security Configuration Guide*.

Additional References

The following sections provide references related to implementing certification authority interoperability.

Related Documents

Related Topic	Document Title
PKI commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Public Key Infrastructure Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5

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, and support for existing RFCs has not been modified 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

Implementing Internet Key Exchange Security Protocol on Cisco IOS XR Software

Internet Key Exchange (IKE) is a key management protocol standard that is used in conjunction with the IP Security (IPSec) standard. IPSec is a feature that provides robust authentication and encryption of IP packets.

IKE is a hybrid protocol that implements the Oakley key exchange and the Skeme key exchange inside the Internet Security Association and Key Management Protocol (ISAKMP) framework. (ISAKMP, Oakley, and Skeme are security protocols implemented by IKE.)

IPSec can be configured without IKE, but IKE enhances IPSec by providing additional features, flexibility, and ease of configuration for the IPSec standard.

This module describes the tasks that you need to implement IKE on your Cisco IOS XR network.

Note For a complete description of the IKE commands used in this chapter, see the *Internet Key Exchange Security Protocol Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing Internet Key Exchange Security Protocol on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.3.0	No modification.
Release 3.4.0	• Support was added for IKE for the Cisco XR 12000 Series Router IPSec VPN SPA. • Support was added to implement IKE for locally sourced and destined traffic on both the Cisco CRS-1 and Cisco XR 12000 Series Router.
Release 3.5.0	• The IP Security VPN Monitoring feature was added. • Banner, Auto-Update, and Browser-Proxy features were added to aid in managing a Cisco Easy VPN remote device. • Pushing a configuration URL through a mode-configuration exchange feature was added.

Contents

- Prerequisites, page SC-20
- Information About Implementing IKE Security Protocol Configurations for IPSec Networks, page SC-20
- Information About IKE for the Cisco IPSec VPN SPA on Cisco IOS XR Software, page SC-32
- How to Implement IKE Security Protocol Configurations for IPSec Networks, page SC-32
- How to Implement IKE for Locally Sourced and Destined Traffic, page SC-58
- How to Implement IKE for Cisco IPSec VPN SPAs on Cisco IOS XR Software, page SC-62
- Configuration Examples for Implementing IKE Security Protocol, page SC-68
- Additional References, page SC-73

Prerequisites

The following prerequisites are required to implement Internet Key Exchange:

- You must be in a user group associated with a task group that includes the proper task IDs for security commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- You must install and activate the package installation envelope (PIE) for the security software.

  For detailed information about optional PIE installation, see the *Cisco IOS XR System Management Guide*.

Information About Implementing IKE Security Protocol Configurations for IPSec Networks

To implement IKE, you should understand the following concepts:

- Supported Standards, page SC-21
- Concessions for Not Enabling IKE, page SC-22
- IKE Policies, page SC-22
- ISAKMP Identity, page SC-26
- ISAKMP Profile Overview, page SC-26
- Mask Preshared Keys, page SC-27
- Preshared Keys Using a AAA Server, page SC-27
- Internet Key Exchange Mode Configuration, page SC-28
- Banner, Auto-Update, and Browser-Proxy, page SC-29
- Pushing a Configuration URL Through a Mode-Configuration Exchange, page SC-29
- Internet Key Exchange Extended Authentication, page SC-30

- Call Admission Control, page SC-30
- Information About IP Security VPN Monitoring, page SC-31

Supported Standards

Cisco implements the following standards:

- IKE—Internet Key Exchange. A hybrid protocol that implements Oakley and Skeme key exchanges inside the ISAKMP framework. IKE can be used with other protocols, but its initial implementation is with the IPSec protocol. IKE provides authentication of the IPSec peers, negotiates IPSec keys, and negotiates IPSec security associations (SAs).

  IKE is implemented following RFC 2409, *The Internet Key Exchange*.

- IPSec—IP Security Protocol. IPSec is a framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPSec provides these security services at the IP layer; it uses IKE to handle negotiation of protocols and algorithms based on local policy and to generate the encryption and authentication keys to be used by IPSec. IPSec is used to protect one or more data flows between a pair of hosts, a pair of security gateways, or a security gateway and a host.

  For more information on IPSec, see the *Implementing IPSec Network Security on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.

- ISAKMP—Internet Security Association and Key Management Protocol. A protocol framework that defines payload formats, the mechanics of implementing a key exchange protocol, and the negotiation of a security association.

  ISAKMP is implemented following the latest version of the *Internet Security Association and Key Management Protocol (ISAKMP)* Internet Draft (RFC 2408).

- Oakley—A key exchange protocol that defines how to derive authenticated keying material.
- Skeme—A key exchange protocol that defines how to derive authenticated keying material, with rapid key refreshment.

The component technologies implemented for use by IKE include the following:

- DES—Data Encryption Standard. An algorithm that is used to encrypt packet data. IKE implements the 56-bit DES-CBC with Explicit IV standard. Cipher Block Chaining (CBC) requires an initialization vector (IV) to start encryption. The IV is explicitly given in the IPSec packet.

  Cisco IOS XR software also implements Triple DES (168-bit) encryption, depending on the software versions available for a specific platform. Triple DES (3DES) is a strong form of encryption that allows sensitive information to be sent over untrusted networks. It enables customers, particularly in the finance industry, to use network-layer encryption.

- AES—Advanced Encryption Standard. 128-bit, 192-bit, and 256-bit AESs are supported.

Note Cisco IOS XR images that have strong encryption (including, but not limited to, 56-bit data encryption feature sets) are subject to U.S. government export controls, and have a limited distribution. Images that are to be installed outside the United States require an export license. Customer orders might be denied or subject to delay because of U.S. government regulations. Contact your sales representative or distributor for more information, or send e-mail to export@cisco.com.

- Diffie-Hellman—A public-key cryptography protocol that allows two parties to establish a shared secret over an unsecure communications channel. Diffie-Hellman is used within IKE to establish session keys. 768-bit, 1024-bit, and 1536-bit Diffie-Hellman groups are supported.
- MD5 (HMAC variant)—Message Digest 5. A hash algorithm used to authenticate packet data. HMAC is a variant that provides an additional level of hashing.
- SHA (HMAC variant)—Secure Hash Algorithm. A hash algorithm used to authenticate packet data. HMAC is a variant that provides an additional level of hashing.
- RSA signatures and RSA encrypted nonces—RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Adelman. RSA signatures provide nonrepudiation, and RSA encrypted nonces provide repudiation. (Repudiation and nonrepudiation are associated with traceability.)

IKE interoperates with the X.509v3 certificates standard. It is used with the IKE protocol when authentication requires public keys. This certificate support allows the 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 to manually specify a shared key at each peer.

Concessions for Not Enabling IKE

IKE is disabled by default in Cisco IOS XR software. If you do not enable IKE, you must make these concessions at the peers:

- You must manually specify all IPSec security associations in the crypto profiles at all peers. (Crypto profile configuration is described in the module *Implementing IPSec Network Security on Cisco IOS XR Software*.)
- The IPSec security associations of the peers never time out for a given IPSec session.
- During IPSec sessions between the peers, the encryption keys never change.
- Antireplay services are not available between the peers.
- Certification authority (CA) support cannot be used.

IKE Policies

You must create IKE policies at each peer. An IKE policy defines a combination of security parameters to be used during the IKE negotiation.

Before you create and configure IKE policies you should understand the following concepts:

- IKE Policy Creation, page SC-23
- Definition of Policy Parameters, page SC-23
- IKE Peer Agreement for Matching Policies, page SC-23
- Value Selection for Parameters, page SC-24
- Policy Creation, page SC-25
- Additional Configuration Required for IKE Policies, page SC-25

IKE Policy Creation

IKE negotiations must be protected, so each IKE negotiation begins by agreement of both peers on a common (shared) IKE policy. This policy states which security parameters will be used to protect subsequent IKE negotiations and mandates how the peers are authenticated.

After the two peers agree on a policy, the security parameters of the policy are identified by a security association established at each peer, and these security associations apply to all subsequent IKE traffic during the negotiation.

You can create multiple, prioritized policies at each peer to ensure that at least one policy matches the policy of a remote peer.

Definition of Policy Parameters

Table 2 lists the five parameters to define in each IKE policy.

Table 2 IKE Policy Parameter Definitions

Parameter	Accepted Values	Keyword	Default Value
Encryption algorithm	56-bit DES-CBC	**des**	56-bit DES-CBC
	168-bit DES	**3des**	
	128-bit AES	**aes**	
	192-bit AES	**aes 192**	
	256-bit AES	**aes 256**	
Hash algorithm	SHA-1 (HMAC variant)	**sha**	SHA-1
	MD5 (HMAC variant)	**md5**	
Authentication method	RSA signatures	**rsa-sig**	RSA signatures
	RSA encrypted nonces	**rsa-encr**	
	Preshared keys	**pre-share**	
Diffie-Hellman group identifier	768-bit Diffie-Hellman or	**1**	768-bit Diffie-Hellman
	1024-bit Diffie-Hellman	**2**	
	1536-bit Diffie-Helman	**5**	
Lifetime of the security association[1]	Any number of seconds	—	86400 seconds (1 day)

1. For information about this lifetime and how it is used, see the command description for the **lifetime** command.

These parameters apply to the IKE negotiations when the IKE security association is established.

IKE Peer Agreement for Matching Policies

When the IKE negotiation begins, IKE looks for an IKE policy that is the same on both peers. The peer that initiates the negotiation will send all its policies to the remote peer, and the remote peer will try to find a match. The remote peer looks for a match by comparing its own highest priority policy against the policies received from the other peer. The remote peer checks each of its policies in order of its priority (highest priority first) until a match is found.

A match is made when both policies from the two peers contain the same encryption, hash, authentication, and Diffie-Hellman parameter values, and when the remote peer's policy specifies a lifetime that is less than or equal to the lifetime in the policy being compared. (If the lifetimes are not identical, the shorter lifetime—from the remote peer's policy—is used.)

If no acceptable match is found, IKE refuses negotiation and IPSec is not established.

If a match is found, IKE completes negotiation, and IPSec security associations are created.

Note Depending on which authentication method is specified in a policy, additional configuration might be required (as described in the "Additional Configuration Required for IKE Policies" section on page 25). If a peer's policy does not have the required companion configuration, the peer does not submit the policy when attempting to find a matching policy with the remote peer.

Value Selection for Parameters

You can select certain values for each parameter, following the IKE standard. But why choose one value over another?

If you are interoperating with a device that supports only one of the values for a parameter, your choice is limited to the value supported by the other device. Aside from this, a trade-off between security and performance often exists, and many of these parameter values represent such a trade-off. You should evaluate the level of security risks for your network and your tolerance for these risks. Then the following tips might help you select which value to specify for each parameter:

- The encryption algorithm has five options: 56-bit DES-CBC, 168-bit DES, 128-bit AES, 192-bit AES, and 256-bit AES.
- The hash algorithm has two options: SHA-1 and MD5.

  MD5 has a smaller digest and is considered to be slightly faster than SHA-1. A demonstrated successful (but extremely difficult) attack has been demonstrated against MD5; however, the HMAC variant used by IKE prevents this attack.
- The authentication method has three options: RSA signatures, RSA encrypted nonces, and preshared keys.
  - RSA signatures provide nonrepudiation for the IKE negotiation (you can prove to a third party after the fact that you did indeed have an IKE negotiation with the remote peer).

    RSA signatures allow the use of a CA. Using a CA can dramatically improve the manageability and scalability of your IPSec network. Additionally, RSA signature-based authentication uses only two public key operations, whereas RAS encryption uses four public key operations, making it costlier in terms of overall performance.

    You can also exchange the public keys manually, as described in the "Manually Configuring RSA Keys" section on page 44.
  - RSA encrypted nonces provide repudiation for the IKE negotiation (you cannot prove to a third party that you had an IKE negotiation with the remote peer).

    RSA encrypted nonces require that peers possess each other's public keys but do not use a certification authority. Instead, two ways exist for peers to get each other's public keys:
  - During configuration, you manually configure RSA keys (as described in the "Manually Configuring RSA Keys" section on page 44).

  - If your local peer has previously used RSA signatures with certificates during a successful IKE negotiation with a remote peer, your local peer already possesses the remote peer's public key. (The peers' public keys are exchanged during the RSA-signatures-based IKE negotiations, if certificates are used.)
  - Preshared keys are clumsy to use if your secured network is large, and they do not scale well with a growing network. However, they do not require use of a certification authority, as do RSA signatures, and might be easier to set up in a small network with fewer than ten nodes. RSA signatures also can be considered more secure when compared with preshared key authentication.

- The Diffie-Hellman group identifier has three options: 768-bit, 1024-bit Diffie-Hellman, and 1536-bit Diffie Helman.

  The 1024-bit Diffie-Hellman and 1536-bit Diffie Helman options are harder to crack but require more CPU time to execute.

- The lifetime of the security association can be set to any value.

  As a general rule, the shorter the lifetime (up to a point), the more secure your IKE negotiations. However, with longer lifetimes, future IPSec security associations can be set up more quickly. For more information about this parameter and how it is used, see the command description for the **lifetime** command.

Policy Creation

You can create multiple IKE policies, each with a different combination of parameter values. For each policy that you create, assign a unique priority (1 through 10,000, with 1 being the highest priority).

You can configure multiple policies on each peer—but at least one of these policies must contain exactly the same encryption, hash, authentication, and Diffie-Hellman parameter values as one of the policies on the remote peer. (The lifetime parameter need not necessarily be the same; see details in the "IKE Peer Agreement for Matching Policies" section on page 23.)

If you do not configure any policies, your router uses the default policy, which is always set to the lowest priority and contains the default value of each parameter.

Additional Configuration Required for IKE Policies

Depending on the authentication method you specify in your IKE policies, you must perform certain additional configuration tasks before IKE and IPSec can successfully use the IKE policies.

Each authentication method requires additional companion configuration as follows:

- RSA signatures method. If you specify RSA signatures as the authentication method in a policy, you may configure the peers to obtain certificates from a CA. (The CA must be properly configured to issue the certificates.) Configure this certificate support as described in the module "Implementing Certification Authority Interoperability."

  The certificates are used by each peer to exchange public keys securely. (RSA signatures require that each peer has the public signature key of the remote peer.) When both peers have valid certificates, they automatically exchange public keys with each other as part of any IKE negotiation in which RSA signatures are used.

  You may also want to exchange the public keys manually, as described in the "Manually Configuring RSA Keys" section on page 44.

- RSA encrypted nonces method. If you specify RSA encrypted nonces as the authentication method in a policy, you must ensure that each peer has the public keys of the other peers.

Unlike RSA signatures, the RSA encrypted nonces method cannot use certificates to exchange public keys. Instead, you ensure that each peer has the others' public keys by one of the following methods:

- Manually configuring RSA keys, as described in the "Manually Configuring RSA Keys" section on page 44.
- Ensuring that an IKE exchange using RSA signatures with certificates has already occurred between the peers. (The peers' public keys are exchanged during the RSA-signatures-based IKE negotiations if certificates are used.)

  To make this happen, specify two policies: a higher-priority policy with RSA encrypted nonces and a lower-priority policy with RSA signatures. When IKE negotiations occur, RSA signatures are used the first time because the peers do not yet have each other's public keys. Then future IKE negotiations are able to use RSA encrypted nonces because the public keys will have been exchanged.

  This alternative requires that you have certification authority support configured.

- Preshared keys authentication method. If you specify preshared keys as the authentication method in a policy, you must configure these preshared keys as described in the "Configuring ISAKMP Preshared Keys in ISAKMP Keyrings" section on page 48.

If RSA encryption is configured and signature mode is negotiated (and certificates are used for signature mode), the peer requests both signature and encryption keys. Basically, the router requests as many keys as the configuration supports. If RSA encryption is not configured, it just requests a signature key.

ISAKMP Identity

You should set the ISAKMP identity for each peer that uses preshared keys in an IKE policy.

When two peers use IKE to establish IPSec security associations, each peer sends its identity to the remote peer. Each peer sends either its hostname or its IP address, depending on how you have set the ISAKMP identity of the router.

By default, the ISAKMP identity of a peer is the IP address of the peer. If appropriate, you could change the identity to be the peer's hostname instead. As a general rule, set the identities of all peers the same way—either all peers should use their IP addresses or all peers should use their host names. If some peers use their host names and some peers use their IP addresses to identify themselves to each other, IKE negotiations could fail if the identity of a remote peer is not recognized and a domain name server (DNS) lookup is unable to resolve the identity.

ISAKMP Profile Overview

The ISAKMP profile is an enhancement to Internet Security Association and Key Management Protocol (ISAKMP) configurations. It enables modularity of ISAKMP configuration for phase 1 negotiations. This modularity allows mapping different ISAKMP parameters to different IP Security (IPSec) tunnels, and mapping different IPSec tunnels to different VPN forwarding and routing (VRF) instances. Currently, many applications and enhancements use the ISAKMP profile, including quality of service (QoS), router certificate management, and Multiprotocol Label Switching (MPLS) VPN configurations.

An ISAKMP profile is a repository for IKE Phase 1 and IKE Phase 1.5 configuration for a set of peers. An ISAKMP profile applies parameters to an incoming IPSec connection identified uniquely through its concept of match identity criteria. These criteria are based on the IKE identity that is presented by incoming IKE connections and includes IP address, fully qualified domain name (FQDN), and group (the Virtual Private Network [VPN] remote client grouping). The granularity of the match identity

criteria imposes the granularity of applying the specified parameters. The ISAKMP profile applies parameters specific to each profile, such as trust points, peer identities, and XAUTH authentication, authorization, and accounting (AAA) list, and so forth.

ISAKMP Profile Considerations

The following considerations are listed on when to use the ISAKMP profile:

- Any router with two or more IPSec connections that requires different phase 1 parameters for different sites (for example, configuring site-to-site and remote access on the same router).
- Custom Internet Key Exchange (IKE) Phase 1 policies might be needed for different peers. For example, whether XAUTH is applied to a specific peer, rather than being applied on every connection.
- IPSec configuration using VRF-aware IPSec, which allows the use of single IP address to connect to different peers with different IKE Phase 1 parameters.

Mask Preshared Keys

A mask preshared key lets a group of remote users with the same level of authentication share an IKE preshared key. The preshared key of the remote peer must match the preshared key of the local peer for IKE authentication to occur.

A mask preshared key is usually distributed through a secure out-of-band channel. In a remote peer-to-local peer scenario, any remote peer with the IKE preshared key configured can establish IKE SAs with the local peer.

If you specify a *subnet-address* value with the **crypto keyring** command, it is up to you to use a subnet address, which allows more peers to share the same key. That is, the preshared key is no longer restricted to use between two users.

Note We do not recommend using 0.0.0.0 as a subnet address, because it encourages group preshared keys, which allow all peers to have the same group key, thereby reducing the security of your user authentication.

Mask preshared keys have the following restrictions:

- The SA cannot be established between the IPSec peers until all IPSec peers are configured for the same preshared key.
- The mask preshared key must be distinctly different for remote users requiring varying levels of authorization. You must configure a new preshared key for each level of trust and assign the correct keys to the correct parties. Otherwise, an untrusted party may obtain access to protected data.

Preshared Keys Using a AAA Server

Preshared keys do not scale well when you are trying to deploy a large scale Virtual Private Network (VPN) without using a CA. When dynamic IP addressing such as DHCP or PPP dialups is used, the changing IP address can make key lookup difficult or impossible unless a mask preshared key is used. However, mask preshared keys are not very secure because a large number of users are given the same secret, thus reducing the security of the secret.

Configuring preshared keys using an authentication, authorization, and accounting (AAA) server allows each user to have his or her own key, which is stored on an external AAA server. This allows for central management of the user database, linking it to an existing AAA database, in addition to allowing every user to have a unique, more secure preshared key.

To configure this feature, you must perform the following tasks at each peer:

- Configure AAA.
- Configure a dynamic crypto ISAKMP profile.
- Configure extended authentication (optional)
- Configure ISAKMP policy.

For information on configuring crypto ISAKMP profiles, including enabling an IPSec peer for preshared keys using an AAA server, see both the "Configuring Crypto Keyrings" section on page 54 and the "Configuring the ISAKMP Profile for Locally Sourced and Destined Traffic" section on page 58. Preshared keys using an AAA server have the following restrictions:

- The shared secret can be accessed only in aggressive mode. The ID of the IKE exchange is used as the username to query AAA if no local key can be found on the Cisco IOS XR router to which the user is trying to connect. Aggressive mode provides the ID in the first part of the IKE exchange; main mode does not provide the ID until the latter part of the IKE exchange, which is too late for key lookup.
- Only the following ID types can be used:
  - IPv4 address (can be different from the one assigned by the Internet service provider [ISP])
  - FQDN (fully qualified domain name)
  - E-mail address

Internet Key Exchange Mode Configuration

IKE mode configuration, as defined by the Internet Engineering Task Force (IETF), allows a gateway to download an IP address (and other network level configuration) to the client as part of an IKE negotiation. Using this exchange, the gateway gives IP addresses to the IKE client to be used as an "inner" IP address encapsulated under IPSec. This method provides a known IP address for the client that can be matched against IPSec policy.

To implement the Cisco IPSec VPN SPAs between remote access clients that have dynamic IP addresses and a corporate gateway, you have to dynamically administer scalable IPSec policy on the gateway once each client is authenticated. With IKE mode configuration, the gateway can set up scalable policy for a very large set of clients irrespective of the IP addresses of those clients.

The client initiation type of IKE mode configuration is supported. The client initiates the configuration mode with the gateway. The gateway responds with an IP address that it has allocated for the client.

Mode configuration must be applied to a crypto ISAKMP profile to be enforced.

For instructions on how to apply mode configuration to a crypto ISAKMP profile, see the "Defining Group Policy Information for Mode Configuration" section on page 36.

Interfaces with crypto ISAKMP profiles, which are configured for IKE mode configuration, may experience a slightly longer connection setup time. This longer setup time is true even for IKE peers that refuse to be configured or do not respond to the configuration mode request. In both cases, the gateway initiates the configuration of the client.

Banner, Auto-Update, and Browser-Proxy

Table 3 describes the features that provide support for attributes that aid in the management of the Cisco Easy VPN remote device.

Table 3 Features that Aid in the Management of the Cisco Easy VPN Remote Device

Feature	Description
Banner	Configures a Cisco Easy VPN server to push a banner to a Cisco Easy VPN remote device.
Auto-Update	Configures a Cisco Easy VPN server to provide an automated mechanism to make software and firmware upgrades automatically available to a Cisco Easy VPN remote device.
Browser-Proxy	Configures a Cisco Easy VPN server so that the Cisco Easy VPN remote device can access resources on the corporate network. With this configuration, the user does not have to manually modify the proxy settings of his or her web browser when connecting and does not have to manually revert the proxy settings when disconnecting.

After a Cisco Easy VPN connection is up, use the **crypto ipsec server send-update** command in EXEC mode to send auto-update notifications at anytime.

Pushing a Configuration URL Through a Mode-Configuration Exchange

When remote devices connect to a corporate gateway for creating an IPsec VPN tunnel, some policy and configuration information must be applied to the remote device when the VPN tunnel is active to allow the remote device to become a part of the corporate VPN. The URL contains the configuration information that the remote device must download and apply to the running configuration.

The configuration that is pushed to the remote device is persistent by default. The configuration is applied when the IPsec tunnel is “up,” but it is not withdrawn when the IPsec tunnel goes “down.” However, it is possible to write a section of configuration that is transient in nature, in which case, the configuration of the section is reverted when the tunnel is disconnected.

There are no restrictions on where the configuration distribution server is physically located. However, we recommended that a secure protocol such as HTTPS (Secure HTTP) be used to retrieve the configuration. The configuration server is located in the corporate network, so because the transfer happens through the IPsec tunnel, insecure access protocols (such as HTTP) are used.

Regarding backward compatibility: the remote device asks for the CONFIGURATION-URL and CONFIGURATION-VERSION attributes. The server sends the configuration url and version attributes whether they were on the group or user. The standard attribute priority scheme, which was used for all the attributes, are also used. There is no built-in restriction to push the configuration, but bootstrap configurations (such as for the IP address) cannot be sent because those configurations are required to set up the Cisco Easy VPN tunnel, and the CONFIGURATION-URL comes into effect only after the Cisco Easy VPN tunnel comes up.

Internet Key Exchange Extended Authentication

IKE extended authentication (Xauth) is a draft RFC based on the IKE protocol. Xauth allows all Cisco IOS XR software AAA authentication methods to perform user authentication in a separate phase after the IKE authentication phase 1 exchange. The AAA configuration list name must match the Xauth configuration list name for user authentication to occur.

Xauth does not replace IKE. IKE allows for device authentication and Xauth allows for user authentication, which occurs after IKE device authentication. Xauth occurs after IKE authentication phase 1 but before IKE IPSec SA negotiation phase 2.

To configure Xauth, perform the following tasks:

- Configure AAA (you must set up an authentication list).
- Configure a static crypto ISAKMP profile.
- Configure ISAKMP policy.
- Configure a dynamic crypto ISAKMP profile (optional).

For information on configuring crypto ISAKMP profiles, see the "Configuring the ISAKMP Profile for Locally Sourced and Destined Traffic" section on page 58.

Call Admission Control

The Call Admission Control (CAC) for Internet Key Exchange (IKE) feature describes the application of CAC to the IKE protocol in Cisco IOS XR software. CAC limits the number of simultaneous IKE security associations (SAs) (that is, calls to CAC) that a router can establish. In addition, there is an option to limit the maximum number of active IKE SAs allowed in the system and the CPU usage that is consumed by the IKE process or global CPU. The main function of CAC is to protect the router from severe resource depletion and to prevent crashes.

IKE Session

You can configure the absolute IKE SA limit by using the **crypto isakmp call admission limit** command. The router drops new IKE SA requests when the value has been reached.

Security Association Limit

A security association (SA) is a description of how two or more entities use security services to communicate securely on behalf of a particular data flow. IKE requires and uses SAs to identify the parameters of its connections. IKE can negotiate and establish its own SA. An IKE SA is used by IKE only, and it is bidirectional. An IKE SA cannot limit IPsec.

IKE drops SA requests based on a user-configured SA limit. To configure an IKE SA limit, use the **crypto isakmp call admission limit** command. When there is a new SA request from a peer router, IKE determines if the number of active IKE SAs plus the number of SAs being negotiated meets or exceeds the configured SA limit. If the number is greater than or equal to the limit, the new SA request is rejected and a system log is generated. This log contains the source destination IP address of the SA request.

Information About IP Security VPN Monitoring

The IP Security (IPSec) VPN Monitoring feature provides VPN session monitoring enhancements that allow you to troubleshoot the Virtual Private Network (VPN) and monitor the end-user interface. Session monitoring includes the following enhancements:

- Ability to specify an Internet Key Exchange (IKE) peer description in the configuration file.
- Summary listing of crypto session status.
- Ability to clear both IKE and IP Security (IPSec) security associations (SAs) using one command-line interface (CLI).
- Ability to expend the filtering mechanism by using the options from the **show crypto session** command.

To implement IPSec VPN monitoring, you must understand the following concepts:

- Crypto Sessions Background, page SC-31
- Per-IKE Peer Description, page SC-31
- Summary Listing of Crypto Session Status, page SC-31
- IKE and IPSec Security Exchange Clear Command, page SC-32

Crypto Sessions Background

A crypto session is a set of IPSec connections (flows) between two crypto endpoints. If the two crypto endpoints use IKE as the keying protocol, they are IKE peers to each other. Typically, a crypto session consists of one IKE security association (for control traffic) and at least two IPSec security associations (for data traffic—one per each direction). There may be duplicated IKE security associations (SAs) and IPSec SAs or duplicated IKE SAs or IPSec SAs for the same session in the duration of rekeying or because of simultaneous setup requests from both sides.

Per-IKE Peer Description

The Per-IKE Peer Description function allows you to enter a description of your choice for an IKE peer. The unique peer description, which includes up to 80 characters, is used whenever you are referencing that particular IKE peer. To add the peer description, use the **description (ISAKMP peer)** command.

The primary application of this description field is for monitoring purposes (for example, when using **show** commands or for logging [syslog messages]). The description field is purely informational.

Summary Listing of Crypto Session Status

You can obtain a list of status information for active crypto sessions by using the **show crypto session** command. The listing includes the following summary status of the crypto session:

- Interface
- IKE SAs that are associated with the peer by whom the IPSec SAs are created
- IPSec SAs serving the flows of a session

Multiple IKE or IPSec SAs can be established for the same peer (for the same session), in which case IKE peer descriptions are repeated with different values for the IKE SAs that are associated with the peer and for the IPSec SAs that are serving the flows of the session.

In addition, you can use the **show crypto session** command with the **detail** keyword to obtain more detailed information about the sessions.

IKE and IPSec Security Exchange Clear Command

The **clear crypto session** command allows you to clear both IKE and IPSec. To clear a specific crypto session or a subset of all the sessions (for example, a single tunnel to one remote site), you need to provide session-specific parameters, such as a local or remote IP address, a local or remote port, a front door VPN routing and forwarding (FVRF) name, or an inside VRF (IVRF) name. Typically, the remote IP address is used to specify a single tunnel to be deleted.

If a local IP address is provided as a parameter when you use the **clear crypto session** command, all the sessions (and their IKE SAs and IPSec SAs) that share the IP address as a local crypto endpoint (IKE local address) are cleared. If you do not provide a parameter, all IPSec SAs and IKE SAs that are in the router are deleted.

Information About IKE for the Cisco IPSec VPN SPA on Cisco IOS XR Software

To implement IKE for the Cisco IPSec VPN SPAs, you should understand the following concept:

- IPSec Dead Peer Detection Periodic Message Option, page SC-32

IPSec Dead Peer Detection Periodic Message Option

The IPSec Dead Peer Detection (DPD) Periodic Message Option feature allows you to configure your router to query the liveliness of its Internet Key Exchange (IKE) peer at regular intervals. The benefit of this approach over the default approach (on-demand dead peer detection) is earlier detection of dead peers.

DPD, which is defined in RFC 3706, is a mechanism used to detect dead IPSec peers. IPSec is a peer-to-peer technology. IP connectivity can be lost between the peers due to routing problems, peer reloading, or some other situation that can result in black holes in which traffic is lost. DPD, based on a traffic-detection method, is one possible mechanism to remedy this situation.

How to Implement IKE Security Protocol Configurations for IPSec Networks

To configure the IKE security protocol for IPSec networks, perform the tasks described in the following sections. The tasks in the first two sections are required; the remaining may be optional, depending on which parameters are configured.

- Enabling or Disabling IKE, page SC-33 (required)
- Configuring IKE Policies, page SC-34 (required)
- Defining Group Policy Information for Mode Configuration, page SC-36 (required)
- Configuring a Banner, page SC-40 (optional)
- Configuring Auto-Upgrade, page SC-40 (optional)

- Configuring a Browser Proxy, page SC-41 (optional)
- Configuring a Browser-Proxy Map to a Group, page SC-42 (optional)
- Configuring the Pushing of a Configuration URL Through a Mode-Configuration Exchange, page SC-43 (optional)
- Manually Configuring RSA Keys, page SC-44 (optional, depending on IKE parameters)
- Configuring ISAKMP Preshared Keys in ISAKMP Keyrings, page SC-48 (optional, depending on IKE parameters)
- Configuring Call Admission Control, page SC-50
- Configuring Crypto Keyrings, page SC-54
- Configuring IP Security VPN Monitoring, page SC-57

Enabling or Disabling IKE

This task enables or disables the Internet Key Exchange protocol.

IKE is disabled by default. IKE need not be enabled for individual interfaces, but it is enabled globally for all interfaces at the router.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp**
3. **no crypto isakmp**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`crypto isakmp`
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp`
		Globally enables IKE at the peer router.

	Command or Action	Purpose
Step 3	**no crypto isakmp** **Example:** `RP/0/RP0/CPU0:router(config)# no crypto isakmp`	(Optional) Disables IKE at the peer router.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IKE Policies

This task configures IKE policies.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp policy** *priority*
3. **encryption {des | 3des | aes | aes 192 | aes 256}**
4. **hash** {**sha** | **md5**}
5. **authentication** {**pre-share** | **rsa-sig** | **rsa-encr**}
6. **group** {**1** | **2** | **5**}
7. **lifetime** *seconds*
8. **end**
   or
   commit
9. **show crypto isakmp policy**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto isakmp policy** ***priority*** **Example:** `RP/0/RP0/CPU0:router(config)# crypto isakmp policy 5`	Identifies the policy to create. • Each policy is uniquely identified by the priority number you assign. • This command places the router in ISAKMP policy configuration mode.
Step 3	**encryption** {**des** | **3des** | **aes** | **aes 192** | **aes 256**} **Example:** `RP/0/RP0/CPU0:router(config-isakmp)# encryption aes`	Specifies the encryption algorithm.
Step 4	**hash** {**sha** | **md5**} **Example:** `RP/0/RP0/CPU0:router(config-isakmp)# hash md5`	Specifies the hash algorithm.
Step 5	**authentication** {**pre-share** | **rsa-sig** | **rsa-encr**} **Example:** `RP/0/RP0/CPU0:router(config-isakmp)# authentication rsa-sig`	Specifies the authentication method.
Step 6	**group** {**1** | **2** | **5**} **Example:** `RP/0/RP0/CPU0:router(config-isakmp)# group 5`	Specifies the Diffie-Hellman group identifier.
Step 7	**lifetime** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-isakmp)# lifetime 50000`	Specifies the lifetime of the security association. The range, in seconds, is from 60 to 86400.

	Command or Action	Purpose

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isakmp)# end`
	or
	`RP/0/RP0/CPU0:router(config-isakmp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**show crypto isakmp policy** **Example:** `RP/0/RP0/CPU0:router# show crypto isakmp policy`	(Optional) Displays all existing IKE policies.

Defining Group Policy Information for Mode Configuration

Although users can belong to only one group for each connection, they may belong to specific groups with different policy requirements. Thus, users may decide to connect to the client using a different group ID by changing their client profile on the VPN device.

This task defines the group policy attributes that are pushed to the client through mode configuration.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp client configuration group** *group-name*
3. **key** *preshared-key*
4. **acl** *acl-name*
5. **backup-server** {*ip-address* | *hostname*}
6. **dns** *primary-server* [*secondary-server*]
7. **domain** *name*
8. **firewall are-u-there**
9. **group-lock**
10. **include-local-lan**

11. **max-logins** *number-of-logins*
12. **max-users** *number-of-users*
13. **netmask** *mask*
14. **pfs**
15. **pool** *name*
16. **save-password**
17. **split-dns** *domain-name*
18. **wins** *primary-server* [*secondary-server*]
19. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto isakmp client configuration group** `group-name` **Example:** `RP/0/RP0/CPU0:router(config)# crypto isakmp client configuration group cisco`	Specifies which group's policy profile is defined and enters ISAKMP group configuration mode. • If no specific group matches and a default group is defined, users are automatically given the default group's policy.
Step 3	**key** *preshared-key* **Example:** `RP/0/RP0/CPU0:router(config-group)# key samplekey`	Specifies the IKE preshared key for group policy attribute definition. **Note** This command *must* be enabled if the client identifies itself with a preshared key.
Step 4	**acl** *acl-name* **Example:** `RP/0/RP0/CPU0:router(config-group)# acl group1`	(Optional) Configures split tunneling. • Use the *acl-name* argument to specify a group of ACL rules that represent protected subnets for split tunneling purposes.
Step 5	**backup-server** {ip-address | hostname} **Example:** `RP/0/RP0/CPU0:router(config-group)# backup-server 10.1.1.1`	Specifies the backup server. • Use the *ip-address* argument to specify the IP address of the server. • Use the *hostname* argument to specify the hostname of the server.

	Command or Action	Purpose
Step 6	**dns** *primary-server* [*secondary-server*] **Example:** `RP/0/RP0/CPU0:router(config-group)# dns 2.2.2.2 2.3.2.3`	Specifies the primary and secondary Domain Name Service (DNS) addresses. • Use the *primary-server* argument to specify the IP address of the primary DNS. • (Optional) Use the *secondary-server* argument to specify the IP address of the secondary DNS.
Step 7	**domain** *name* **Example:** `RP/0/RP0/CPU0:router(config-group)# domain cisco.com`	Specifies the DNS domain to which a group belongs. • Use the *name* argument to specify the default name of the DNS domain.
Step 8	**firewall are-u-there** **Example:** `RP/0/RP0/CPU0:router(config-group)# firewall are-u-there`	Adds the Firewall-Are-U-There attribute to the server group if your PC is running the Black Ice or Zone Alarm personal firewalls.
Step 9	**group-lock** **Example:** `RP/0/RP0/CPU0:router(config-group)# group-lock`	Allows you to enter your extended authentication (Xauth) username, including the group name, when preshared key authentication is used with IKE.
Step 10	**include-local-lan** **Example:** `RP/0/RP0/CPU0:router(config-group)# include-local-lan`	Configures the Include-Local-LAN attribute to allow a nonsplit-tunneling connection to access the local subnetwork at the same time as the client.
Step 11	**max-logins** *number-of-logins* **Example:** `RP/0/RP0/CPU0:router(config-group)# max-logins 8`	Specifies the maximum number of concurrent logins that are allowed for a certain user. • Use the *number-of-logins* argument to specify the number of logins. The value ranges from 0 to 16 and 384.
Step 12	**max-users** *number-of-users* **Example:** `RP/0/RP0/CPU0:router(config-group)# max-users 1200`	Limits the number of connections to a specific server group. • Use the *number-of-users* argument to specify the number of connected users. The value ranges from 0 to 16 and 384.
Step 13	**netmask** *mask* **Example:** `RP/0/RP0/CPU0:router(config-group)# netmask 255.255.255.0`	Sets the IP network mask. • Use the *mask* argument to specify the IP network mask.
Step 14	**pfs** **Example:** `RP/0/RP0/CPU0:router(config-group)# pfs`	Configures a server to notify the client of the central-site policy regarding whether PFS is required for any IP Security (IPSec) Security Association (SA).

	Command or Action	Purpose
Step 15	**pool** *name* **Example:** `RP/0/RP0/CPU0:router(config-group)# pool dog`	Defines the name of an address-pool in which an address is allocated if required. • Use the *name* argument to specify the name of the local pool address.
Step 16	**save-password** **Example:** `RP/0/RP0/CPU0:router(config-group)# save-password`	Saves your extended authentication (Xauth) password locally on your PC or Easy VPN client.

Step 17	**split-dns** *domain-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config-group)# split-dns green.com`
	`RP/0/RP0/CPU0:router(config-group)# split-dns acme.org`
		Specifies a domain name that must be tunneled or resolved to the private network.
		• Use the *domain-name* argument to specify the name of the DNS domain that must be tunneled or resolved to the private network.
		Note If you have to configure more than one domain name, you have to add a **split-dns** command line for each.

Step 18	**wins** *primary-server* [*secondary-server*] **Example:** `RP/0/RP0/CPU0:router(config-group)# wins 10.1.1.2 10.1.1.3`	Specifies the primary and secondary Windows Internet Naming Service (WINS) servers. • Use the *primary-server* argument to specify the name of the primary WINS server. • (Optional) Use the *secondary-server* argument to specify the name of the secondary WINS server.

Step 19	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-group)# end`
	or
	`RP/0/RP0/CPU0:router(config-group)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Banner

This task describes how to configure a banner for a Cisco Easy VPN server.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp client configuration group** *group-name*
3. **banner** {*banner-text*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp client configuration group** group-name
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp client configuration group cisco`
	`RP/0/RP0/CPU0:router(config-group)#`
		Specifies which group's policy profile is defined and enters ISAKMP group configuration mode.
		• If no specific group matches, you are automatically given the default group's policy.
		• The default group is also used for the other attributes so they must also be checked and updated.

Step 3	**banner** {*banner-text*} **Example:** `RP/0/RP0/CPU0:router(config-group)# banner thequickbrowndog`	Specifies the text of the banner.

Configuring Auto-Upgrade

This task describes how to configure automatic update parameters for a Cisco Easy VPN remote device.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp client configuration group** *group-name*
3. **auto-update client** {*type-of-system*} {**url** *url*} {**rev** *review-version*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp client configuration group** group-name
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp client configuration group cisco`
	`RP/0/RP0/CPU0:router(config-group)#`
		Specifies which group's policy profile is defined and enters ISAKMP group configuration mode.
		• If no specific group matches, you are automatically given the default group's policy.
		• The default group is also used for the other attributes so they must also be checked and updated.

Step 3	**auto-update client** {*type-of-system*} {**url** *url*} {**rev** *review-version*} **Example:** `RP/0/RP0/CPU0:router(config-group)# auto-update client Win2000 url http:www.ourcompanysite.com/newclient rev 3.0.1`	Configures automatic update parameters for a Cisco Easy VPN remote device. The example shows that the update parameters are set for a Windows 2000 operating system, a URL of http:www.ourcompanysite.com/newclient, and version 3.0.1. For a list of the reserved names that the Cisco Easy VPN client expects for each type of operating system, which are used for the *type-of-system* argument, see *Cisco IOS XR System Security Command Reference*.

Configuring a Browser Proxy

This task describes how to configure browser-proxy parameters for a Cisco Easy VPN remote device.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp client configuration browser-proxy** {*browser-proxy-name*}
3. **proxy** {**auto-detect | bypass-local | exception-list** *semicolon delimited exception list* **| none | server**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp client configuration browser-proxy** {*browser-proxy-name*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp client configuration browser-proxy bproxy`
	`RP/0/RP0/CPU0:router(config-crypto-isakmp-browser-proxy)#`
		Configures browser-proxy parameters for a Cisco Easy VPN remote device and enters ISAKMP browser proxy configuration mode.

Step 3	**proxy** {**auto-detect** | **bypass-local** | **exception-list** *semicolon delimited exception list* | **none** | **server**} **Example:** `RP/0/RP0/CPU0:router(config-crypto-isakmp-browser-proxy)# proxy auto-detect`	Configures proxy parameters for a Cisco Easy VPN remote device.

Configuring a Browser-Proxy Map to a Group

This task configures a browser-proxy map to a group.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp client configuration group** *group-name*
3. **browser-proxy** {*browser-proxy-map-name*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp client configuration group** *group-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp client configuration group cisco`
	`RP/0/RP0/CPU0:router(config-group)#`
		Specifies which group's policy profile is defined and enters ISAKMP group configuration mode.
		• If no specific group matches, you are automatically given the default group's policy.
		• The default group is also used for the other attributes so they must also be checked and updated.

Step 3	**browser-proxy** {*browser-proxy-map-name*} **Example:** `RP/0/RP0/CPU0:router(config-group)# browser-proxy EZVPN`	Specifies browser-proxy parameter settings to a group.

Configuring the Pushing of a Configuration URL Through a Mode-Configuration Exchange

This task configures a Cisco Easy VPN server to push a configuration URL through a Mode-Configuration Exchange.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp client configuration group** *group-name*
3. **configuration url** {*url*}
4. **configuration version** {*version-number*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp client configuration group** *group-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp client configuration group cisco`
	`RP/0/RP0/CPU0:router(config-group)#`
		Specifies which group's policy profile is defined and enters ISAKMP group configuration mode.
		• If no specific group matches, you are automatically given the default group's policy.
		• The default group is also used for the other attributes so they must also be checked and updated.

Step 3	**configuration url** {*url*} **Example:** `RP/0/RP0/CPU0:router(config-group)# configuration url http://10.10.8.8/easy.cfg`	Specifies the URL the remote device must use to get the configuration from the server.
Step 4	**configuration version** {*version-number*} **Example:** `RP/0/RP0/CPU0:router(config-group)# configuration version 10`	Specifies the version of the configuration. • The *version-number* argument is an unsigned integer in the range of 1 to 10.

Manually Configuring RSA Keys

Manually configure RSA keys when you specify RSA encrypted nonces as the authentication method in an IKE policy and you are not using a CA.

To manually configure RSA keys, perform these tasks at each IPSec peer that uses RSA encrypted nonces in an IKE policy:

- RSA Keys Generation, page SC-44
- Configuring ISAKMP Identity, page SC-44
- Configuring RSA Public Keys of All the Other Peers, page SC-46

RSA Keys Generation

For instructions on how to generate RSA keys, see the "Generating an RSA Key Pair" section on page 7 in the *Implementing Certification Authority Interoperability on Cisco IOS XR Software* module.

Configuring ISAKMP Identity

This task configures the ISAKMP identity of a peer.

Remember to repeat these tasks at each peer that uses preshared keys in an IKE policy.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp identity** {**address** | **hostname**}
3. **host** *hostname address1* [*address2...address8*]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto isakmp identity** {**address** | **hostname**} **Example:** `RP/0/RP0/CPU0:router(config)# crypto isakmp identity address`	(At the local peer) Specifies the peer's ISAKMP identity by IP address or by hostname. See the **crypto isakmp identity** command description for guidelines for when to use the IP address and when to use the hostname.
Step 3	**host** ***hostname address1*** [***address2...address8***] **Example:** `RP/0/RP0/CPU0:router(config)# host host1 10.0.0.5`	(At all remote peers) Maps the peer's hostname to its IP addresses at all the remote peers. • This command is used if the local peer's ISAKMP identity was specified using a hostname. • This step might be unnecessary if the hostname or address is already mapped in a DNS server.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring RSA Public Keys of All the Other Peers

This task configures the RSA public keys of all the other peers.

Remember to repeat these tasks at each peer that uses RSA encrypted nonces in an IKE policy.

SUMMARY STEPS

1. **configure**
2. **crypto keyring** *keyring-name* [**vrf** *fvrf-name*]
3. **rsa-pubkey** {**address** *address* | **name** *fqdn*} [**encryption** | **signature**]
4. **address** *ip-address*
5. **key-string** *key-string*
6. **quit**
7. **end**
   or
   commit
8. **show crypto key pubkey-chain rsa** [**name** *key-name* | **address** *key-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto keyring** *keyring-name* [**vrf** *fvrf-name*]
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto keyring vpnkeyring`
	`RP/0/RP0/CPU0:router(config-keyring)#`
		Defines a crypto keyring during IKE authentication
		• Enters keyring configuration mode.
		• Use the *keyring-name* argument to specify the name of the crypto keyring.
		• (Optional) Use the **vrf** keyword to specify that the front door virtual routing and forwarding (FVRF) name is the keyring that is referenced.

	Command or Action	Purpose
Step 3	**rsa-pubkey** {**address** *address* | **name** *fqdn*} [**encryption** | **signature**] **Example:** `RP/0/RP0/CPU0:router(config-keyring)# rsa-pubkey name host.vpn.com RP/0/RP0/CPU0:router(config-pubkey)`	Defines the Rivest, Shamir, and Adelman (RSA) manual key to be used for encryption or signature during Internet Key Exchange (IKE) authentication. • Use the **address** keyword to specify the IP address of the RSA public key of the remote peer. The address argument is the IP address of the remote RSA public key of the remote peer that you manually configure. • Use the **name** keyword to specify the fully qualified domain name (FQDN) of the peer. • Use the **encryption** keyword to specify that the key is used for encryption. • Use the **signature** keyword to specify that the key is used for a signature. The **signature** keyword is the default.
Step 4	**address** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-pubkey)# address 10.5.5.1`	Specifies the remote peer's IP address. • You can use this command if you used a fully qualified domain name to name the remote peer.
Step 5	**key-string** ***key-string*** **Example:** `RP/0/RP0/CPU0:router(config-pubkey)# key-string 005C300D 06092A86 4886F70D 01010105 ...`	Specifies the remote peer's RSA public key. • This is the key previously displayed by the remote peer's administrator when the remote router's RSA keys were generated. • To avoid mistakes, you should cut and paste the key data (instead of attempting to enter in the data). • Enter a key on each line. You must enter the **key-string** command before the key. • When you have finished specifying the remote peer's RSA key, return to global configuration mode by entering **quit** at the public key configuration prompt.
Step 6	**quit** **Example:** `RP/0/RP0/CPU0:router(config-pubkey)# quit`	Returns to global configuration mode.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show crypto pubkey-chain rsa** [**name** ***key-name*** | **address** ***key-address***] **Example:** `RP/0/RP0/CPU0:router# show crypto pubkey-chain rsa`	(Optional) Displays a list of all the RSA public keys stored on your router. • Use the optional name or address keyword to display details about a particular RSA public key stored on your router.

Configuring ISAKMP Preshared Keys in ISAKMP Keyrings

This task configures ISAKMP preshared keys in ISAKMP keyrings.

Prerequisites

To configure ISAKMP preshared keys in ISAKMP keyrings, perform these tasks at each peer that uses preshared keys in an IKE policy:

- Set the ISAKMP identity of each peer. Each peer's identity should be set either to its hostname or by its IP address. By default, a peer's identity is set to its IP address. Setting ISAKMP identities is described in the "Configuring ISAKMP Identity" section on page 44.
- Specify the shared keys at each peer. Note that a given preshared key is shared between two peers. At a given peer you could specify the same key to share with multiple remote peers; however, a more secure approach is to specify different keys to share between different pairs of peers.

You must specify the support for masked preshared keys.

Remember to repeat these tasks at each peer that uses preshared keys in an IKE policy.

SUMMARY STEPS

1. **configure**
2. **crypto keyring** *keyring-name* [**vrf** *fvrf-name*]
3. **pre-shared-key** {**address** *address* [*mask*] | **hostname** *hostname*} **key** *key*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto keyring** *keyring-name* [**vrf** *fvrf-name*]
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto keyring vpnkeyring`
	`RP/0/RP0/CPU0:router(config-keyring)#`
		Defines a crypto keyring during IKE authentication.
		• Use the *keyring-name* argument to specify the name of the crypto keyring.
		• (Optional) Use the **vrf** keyword to specify that the front door virtual routing and forwarding (FVRF) name is the keyring that is referenced.

	Command or Action	Purpose

Step 3	**pre-shared-key** {**address** *address* [*mask*] | **hostname** *hostname*} **key** *key*
	
	Example:
	`RP/0/RP0/CPU0:router(config-keyring)# pre-shared-key address 10.72.23.11 key vpnkey`
	`RP/0/RP0/CPU0:router(config-keyring)# pre-shared-key hostname mycisco.com key vpnkey`
		Defines a preshared key for IKE authentication.
		• Use the **address** keyword to specify the IP address of the remote peer or a subnet and mask.
		• (Optional) Use the *mask* argument to match the range of the address.
		• Use the **hostname** keyword to specify the fully qualified domain name (FQDN) of the peer.
		• (Optional) Use the **key** keyword to specify the key.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-keyring)# end`
	or
	`RP/0/RP0/CPU0:router(config-keyring)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Call Admission Control

These tasks are used to configure Call Admission Control (CAC):

- Configuring the IKE Security Association Limit, page SC-50
- Configuring the System Resource Limit, page SC-52

Configuring the IKE Security Association Limit

This task configures the IKE security admission limit.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp call admission limit** {**in-negotiation-sa** *number* | **sa** *number*}

3. **end**
   or
   commit
4. **show crypto isakmp call admission statistics**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto isakmp call admission limit** {**in-negotiation-sa** *number* | **sa** *number*} **Example:** `RP/0/RP0/CPU0:router(config)# crypto isakmp call admission limit sa 25`	Specifies the maximum number of IKE SAs that the router can establish before IKE begins rejecting new SA requests. • Use the **in-negotiation-sa** keyword to specify the maximum number of in-negotiation (embryonic) IKE security associations (SAs) that the router can establish before IKE begins rejecting new SA requests. The range for the number argument is from 1 to 100000. • Use the **sa** keyword to specify the maximum number of active IKE SAs that the router can establish. The range for the *number* argument is from 1 to 100000.

	Command or Action	Purpose

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 4	**show crypto isakmp call admission statistics** **Example:** `RP/0/RP0/CPU0:router# show crypto isakmp call admission statistics`	Monitors crypto CAC statistics.

Configuring the System Resource Limit

This task configures the system resource limit.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp call admission limit** {**cpu** {**total** *percent* | **ike** *percent*}}
3. **end**
   or
   commit
4. **show crypto isakmp call admission statistics**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto isakmp call admission limit** {**cpu** {**total** *percent* | **ike** *percent*}} **Example:** `RP/0/RP0/CPU0:router(config)# crypto isakmp call admission limit cpu total 90`	Specifies the maximum number of IKE SAs that the router can establish before IKE begins rejecting new SA requests. • Use the **cpu** keyword to specify the total resource limit for the CPU usage. • Use the **total** keyword to specify the maximum total CPU usage to accept new calls. The range for the *percent* argument is from 1 to 100. • Use the **ike** keyword to specify the maximum IKE CPU usage to accept new calls. The range for the percent argument is from 1 to 100.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 4	**show crypto isakmp call admission statistics** **Example:** `RP/0/RP0/CPU0:router# show crypto isakmp call admission statistics`	Monitors crypto CAC statistics.

Configuring Crypto Keyrings

A crypto keyring is a repository of preshared and Rivest, Shamir, and Adelman (RSA) public keys. The router can have zero or more keyrings. Each keyring optionally allows the specification of a VRF in which the keys defined in the keyring belong.

This task configures crypto keyrings.

Crypto Keyrings Configuration Guidelines and Restrictions

Follow these guidelines and restrictions when configuring crypto keyrings:

- The VRF associated with a crypto keyring cannot be changed. A different keyring must be configured with the new VRF value.
- Address overlapping in a keyring is not allowed and must be enforced during configuration.
- A crypto keyring is attached to one or more ISAKMP profiles and cannot be deleted while in use.

SUMMARY STEPS

1. **configure**
2. **crypto keyring** *keyring-name* [**vrf** *fvrf-name*]
3. **description** *string*
4. **local-address** *ip-address*
5. **pre-shared-key** {**address** *address* [*mask*] | **hostname** *hostname*} **key** *key*
6. **rsa-pubkey** {**address** *address* | **name** *fqdn*} [**encryption** | **signature**]
7. **key-string** *key-string*
8. **quit**
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto keyring** *keyring-name* [**vrf** *fvrf-name*] **Example:** `RP/0/RP0/CPU0:router(config)# crypto keyring vpnkey`	Defines a crypto keyring to be used during IKE authentication. • Use the *keyring-name* argument as the name of the crypto keyring. • Use the **vrf** keyword to specify that the front door virtual routing and forwarding (FVRF) name is the keyring that is referenced. The *fvrf-name* argument must match the FVRF name that was defined during a (VRF) configuration.
Step 3	**description** *string* **Example:** `RP/0/RP0/CPU0:router(config-keyring# description this is a sample keyring`	Creates a one-line description for a keyring. • Use the *string* argument to specify the character string that describes the keyring.
Step 4	**local-address** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-keyring)# local-address 130.40.1.1`	Limits the scope of an ISAKMP keyring configuration to a local termination address or interface. • Use the *ip-address* argument to specify the IP address to which to bind.
Step 5	**pre-shared-key** {**address** *address* [*mask*] | **hostname** *hostname*} **key** *key* **Example:** `RP/0/RP0/CPU0:router(config-keyring)# pre-shared-key address 10.72.23.11 key vpnkey`	Defines a preshared key to be used for IKE authentication. • Use the **address** keyword to specify the IP address of the remote peer or a subnet and mask. The *mask* argument is optional. • Use the **hostname** keyword to specify the fully qualified domain name (FQDN) of the peer. • Use the **key** keyword to specify the secret.

	Command or Action	Purpose
Step 6	**rsa-pubkey** {**address** *address* | **name** *fqdn*} [**encryption** | **signature**] **Example:** `RP/0/RP0/CPU0:router(config-keyring)# rsa-pubkey name host.vpn.com` `RP/0/RP0/CPU0:router(config-keyring)# rsa-pubkey name host.vpn.com`	Defines a Rivest, Shamir, and Adelman (RSA) public key by address or hostname. • Use the **address** keyword to specify the IP address of the RSA public key of the remote peer. The *address* argument is the IP address of the remote RSA public key of the remote peer that you manually configure. • Use the **name** keyword to specify the FQDN of the peer. • (Optional) Use the **encryption** keyword to specify that the key is used for encryption. • (Optional) Use the **signature** keyword to specify that the key is used for a signature. The **signature** keyword is the default.
Step 7	**key-string** *key-string* **Example:** `RP/0/RP0/CPU0:router(config-pubkey)# key-string 005C300D 06092A86 4886F70D 01010105`	Manually specifies the RSA public key of a remote peer.
Step 8	**quit** **Example:** `RP/0/RP0/CPU0:router(config-pubkey)# quit`	Returns to global configuration mode.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IP Security VPN Monitoring

The following sections describe how to configure IP Security (IPSec) VPN monitoring:

- Adding the Description of an IKE Peer, page SC-57 (optional)
- Clearing a Crypto Session, page SC-58 (optional)

Adding the Description of an IKE Peer

This task allows you to add the description of an IKE peer to an IPSec VPN session.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp peer** {**address** *ip-address* | **hostname** *hostname*} [**description** *line* | **vrf** *fvrf-name*]
3. **description** *line-of-description*
4. **end**
   or
   commit
5. **show crypto isakmp peers** [*ip-address* | **vrf** *vrf-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp peer** {**address** *ip-address* | **hostname** *hostname*} [**description** *line* | **vrf** *fvrf-name*]
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp peer address 40.40.40.2`
	`RP/0/RP0/CPU0:router(config-isakmp-peer)`
		Enables an IPSec peer for IKE querying of authentication, authorization, and accounting (AAA) for tunnel attributes in aggressive mode and enters ISAKMP peer configuration mode.

Step 3	**description** *line-of-description* **Example:** `RP/0/RP0/CPU0:router(config-isakmp-peer)# description citeA`	Adds a description for an IKE peer.

	Command or Action	Purpose

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isakmp-peer)# end`
	or
	`RP/0/RP0/CPU0:router(config-isakmp-peer)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show crypto isakmp peers** [*ip-address* | **vrf** *vrf-name*] **Example:** `RP/0/RP0/CPU0:router# show crypto isakmp peers`	Displays peer descriptions.

Clearing a Crypto Session

Use the **clear crypto session** command in EXEC mode to delete the crypto sessions (IP Security [IPSec] and Internet Key Exchange [IKE] security associations [SAs]) for users and groups.

How to Implement IKE for Locally Sourced and Destined Traffic

This section contains the following procedure:

- Configuring the ISAKMP Profile for Locally Sourced and Destined Traffic, page SC-58

Configuring the ISAKMP Profile for Locally Sourced and Destined Traffic

An ISAKMP profile is a repository of commands for a set of peers.

This task configures the ISAKMP profile for locally sourced and destined traffic.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp profile** [**local**] *profile-name*
3. **description** *string*
4. **keepalive disable**
5. **self-identity** {**address** | **fqdn** | **user-fqdn** *user-fqdn*}
6. **keyring** *keyring-name*
7. **match identity** {**group** *group-name* | **address** *address* [*mask*] **vrf** [*fvrf*] | **host** *hostname* | **host domain** *domain-name* | **user** *username* | **user domain** *domain-name*}
8. **set interface tunnel-ipsec** *intf-index*
9. **set ipsec-profile** *profile-name*
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp profile** [**local**] *profile-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp profile local vpnprofile`
	`RP/0/RP0/CPU0:router(config-isa-prof)#`
		Defines an ISAKMP profile and audits IPSec user sessions.
		• (Optional) Use the **local** keyword to specify that the profile is used for locally sourced or terminated traffic.
		Note The **local** keyword is specific only to the Cisco IPSec VPN SPA.
		• Use the *profile-name* argument to specify the name of the user profile.

Step 3	**description** *string* **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# description this is a sample profile`	Creates a description for a keyring. • Use the *string* argument to specify the character string that describes the keyring.
Step 4	**keepalive disable** **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# keepalive disable`	Lets the gateway send DPD messages to the Cisco IOS XR peer. • Use the **disable** keyword to disable the keepalive global declarations.

	Command or Action	Purpose
Step 5	**self-identity** {**address** | **fqdn** | **user-fqdn** *user-fqdn*} **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# self-identity user-fqdn user@vpn.com`	Defines the identity that the local IKE uses to identify itself to the remote peer. • Use the **address** keyword to specify the IP address of the local endpoint. • Use the **fqdn** keyword to specify the fully qualified domain name (FQDN) of the host. • Use the **user-fqdn** keyword to specify that the user FQDN was sent to the remote endpoint.
Step 6	**keyring** *keyring-name* **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# keyring vpnkeyring`	Configures a keyring with an ISAKMP profile. • Use the *keyring-name* argument to specify the keyring name, which must match the keyring name that was defined in the global configuration.

	Command or Action	Purpose

Step 7	**match identity** {**group** *group-name* | **address** *address* [*mask*] **vrf** [*fvrf*] | **host** *hostname* | **host domain** *domain-name* | **user** *username* | **user domain** *domain-name*}
	
	Example:
	`RP/0/RP0/CPU0:router(config-isa-prof)# match identity group vpngroup`
	`RP/0/RP0/CPU0:router(config-isa-prof-match)#`
		Matches the identity from a peer in an ISAKMP profile.
		• Use the **group** keyword to specify a Unity group that matches identification (ID) type ID_KEY_ID. If RSA signatures are used, the *group-name* argument matches the organizational unit (OU) field of the distinguished name (DN).
		• Use the **address** keyword to match the *address* argument with the ID type ID_IPV4_ADDR.
		• Use the *mask* argument to specify a range of addresses.
		• Use the **vrf** keyword to specify the front door VPN routing and forwarding (VRF) of the peer.
		• Use the *fvrf* argument to match the address in the front door virtual router forwarding (FVRF) Virtual Private Network (VPN) space.
		• Use the **host** keyword to specify an identity that matches the type ID_FQDN, whose fully qualified domain name (FQDN) ends with the domain name.
		• Use the **host domain** keyword to specify an identity that matches type ID_FQDN. The domain name is the same as the *domain-name* argument.
		• Use the **user** keyword to specify an identity that matches the FQDN.
		• Use the **user domain** keyword to specify an identity that matches the type ID_USER_FQDN. When the **user domain** keyword is present, all users having identities of the type ID_USER_FQDN and ending with *domain-name* are matched.

Step 8	**set interface tunnel-ipsec** *intf-index* **Example:** `RP/0/RP0/CPU0:router(config-isa-prof-match)# set interface tunnel-ipsec 50`	Predefines the interface instance when IKE negotiates for IPSec service associations (SAs) for the traffic that is locally sourced or terminated and the local endpoint is the IKE responder. • Use the *intf-index* argument to set the range from 0 to 4294967295.

	Command or Action	Purpose
Step 9	**set ipsec-profile** *profile-name* **Example:** `RP/0/RP0/CPU0:router(config-isa-prof-match)# set ipsec-profile myprofile`	Predefines the IPSec profile instance when IKE negotiates for IPSec service associations (SAs) for the traffic that is locally sourced or terminated and the local endpoint is the IKE responder. • Use the *profile-name* argument to set the name of the IPSec profile.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isa-prof-match)# end`
	or
	`RP/0/RP0/CPU0:router(config-isa-prof-match)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

How to Implement IKE for Cisco IPSec VPN SPAs on Cisco IOS XR Software

This section contains the following procedures:

- Configuring a Periodic Dead Peer Detection Message, page SC-63
- Configuring the ISAKMP Profile for Service Interfaces, page SC-64

Configuring a Periodic Dead Peer Detection Message

This task configures a periodic dead peer detection (DPD) message.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp keepalive** *seconds retry-seconds* [**periodic** | **on-demand**]
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**crypto isakmp keepalive** *seconds retry-seconds* [**periodic** | **on-demand**] **Example:** `RP/0/RP0/CPU0:router(config)# crypto isakmp keepalive 20 20 on-demand`	Uses the IKE security association (SA) feature to provide a mechanism to detect loss of connectivity between two IP Security (IPSec) peers. • Use the *seconds* argument to specify the number of seconds between keepalive messages. The range is from 10 to 3600. • Use the *retry-seconds* argument to specify the number of seconds between retries if keepalive fails. The range is from 2 to 60. • (Optional) Use the **periodic** keyword to specify the keepalive messages that are sent at regular intervals for DPD messages. • (Optional) Use the **on-demand** keyword to specify the DPD retries that are sent on demand.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the ISAKMP Profile for Service Interfaces

This task configures the ISAKMP profile for service interfaces.

SUMMARY STEPS

1. **configure**
2. **crypto isakmp profile** [**local**] *profile-name*

3. **description** *string*
4. **keepalive disable**
5. **self-identity** {**address** | **fqdn** | **user-fqdn** *user-fqdn*}
6. **keyring** *keyring-name*
7. **match identity** {**group** *group-name* | **address** *address* [*mask*] **vrf** [*fvrf*] | **host** *hostname* | **host domain** *domain-name* | **user** *username* | **user domain** *domain-name*}
8. **set interface** {**service-ipsec** | **service-gre**} *intf-index*
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto isakmp profile** [**local**] *profile-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto isakmp profile vpnprofile`
	`RP/0/RP0/CPU0:router(config-isa-prof)#`
		Defines an ISAKMP profile and audits IPSec user sessions.
		• (Optional) Use the **local** keyword to specify that the profile is used for locally sourced or terminated traffic.
		Note The **local** keyword is specific only to the Cisco IPSec VPN SPA.
		• Use the *profile-name* argument to specify the name of the user profile.

Step 3	**description** *string* **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# description this is a sample profile`	Creates a description for a keyring. • Use the *string* argument to specify the character string that describes the keyring.
Step 4	**keepalive disable** **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# keepalive disable`	Lets the gateway send DPD messages to the Cisco IOS XR peer. • Use the **disable** keyword to disable the keepalive global declarations.

	Command or Action	Purpose
Step 5	**self-identity** {**address** | **fqdn** | **user-fqdn** *user-fqdn*} **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# self-identity user-fqdn user@vpn.com`	Defines the identity that the local IKE uses to identify itself to the remote peer. • Use the **address** keyword to specify the IP address of the local endpoint. • Use the **fqdn** keyword to specify the fully qualified domain name (FQDN) of the host. • Use the **user-fqdn** keyword to specify that the user FQDN was sent to the remote endpoint.
Step 6	**keyring** *keyring-name* **Example:** `RP/0/RP0/CPU0:router(config-isa-prof)# keyring vpnkeyring`	Configures a keyring with an ISAKMP profile. • Use the *keyring-name* argument to specify the keyring name, which must match the keyring name that was defined in the global configuration.

	Command or Action	Purpose

Step 7	**match identity** {**group** *group-name* | **address** *address* [*mask*] **vrf** [*fvrf*] | **host** *hostname* | **host domain** *domain-name* | **user** *username* | **user domain** *domain-name*}
	
	Example:
	`RP/0/RP0/CPU0:router(config-isa-prof)# match identity group vpngroup`
	`RP/0/RP0/CPU0:router(config-isa-prof-match)#`
		Matches the identity from a peer in an ISAKMP profile.
		• Use the **group** keyword to specify a Unity group that matches identification (ID) type ID_KEY_ID. If RSA signatures are used, the *group-name* argument matches the organizational unit (OU) field of the distinguished name (DN).
		• Use the **address** keyword to match the *address* argument with the ID type ID_IPV4_ADDR.
		• Use the *mask* argument to specify a range of addresses.
		• Use the **vrf** keyword to specify the front door VPN routing and forwarding (VRF) of the peer.
		• Use the *fvrf* argument to match the address in the front door virtual router forwarding (FVRF) Virtual Private Network (VPN) space.
		• Use the **host** keyword to specify an identity that matches the type ID_FQDN, whose fully qualified domain name (FQDN) ends with the domain name.
		• Use the **host domain** keyword to specify an identity that matches type ID_FQDN. The domain name is the same as the *domain-name* argument.
		• Use the **user** keyword to specify an identity that matches the FQDN.
		• Use the **user domain** keyword to specify an identity that matches the type ID_USER_FQDN. When the **user domain** keyword is present, all users having identities of the type ID_USER_FQDN and ending with *domain-name* are matched.

	Command or Action	Purpose

Step 8	**set interface** {**service-ipsec** | **service-gre**} *intf-index*
	
	Example:
	`RP/0/RP0/CPU0:router(config-isa-prof-match)# set interface service-ipsec 50`
	or
	`RP/0/RP0/CPU0:router(config-isa-prof-match)# set interface service-gre 1000`
		Predefines the virtual interface when IKE negotiates for IPSec SAs and the local endpoint is the IKE responder.
		• Use the **service-ipsec** keyword to specify the IPSec service interfaces.
		• Use the **service-gre** keyword to specify the GRE service interfaces.
		• Use the *intf-index* argument to set the range from 1 to 65535.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isa-prof-match)# end`
	or
	`RP/0/RP0/CPU0:router(config-isa-prof-match)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing IKE Security Protocol

This section provides the following configuration examples:

- Creating IKE Policies: Example, page SC-69
- Configuring a service-ipsec Interface with a Dynamic Profile: Example, page SC-69
- Configuring Easy VPN with a Local AAA: Example, page SC-70
- Configuring VRF-Aware: Example, page SC-71

Creating IKE Policies: Example

This example shows how to create two IKE policies with policy 15 as the highest priority, policy 20 as the next priority, and the existing default priority as the lowest priority.

```
crypto isakmp policy 15
    encryption 3des
    hash md5
    authentication rsa-sig
    group 2
    lifetime 5000
crypto isakmp policy 20
    authentication pre-share
    lifetime 10000
```

In the example, the **encryption des** of policy 15 would not appear in the written configuration because this is the default value for the encryption algorithm parameter.

If the **show crypto isakmp policy** command is issued with this configuration, the output is as follows:

```
Protection suite priority 15
    encryption algorithm:DES - Data Encryption Standard (56 bit keys)
    hash algorithm:Message Digest 5
    authentication method:Rivest-Shamir-Adelman Signature
    Diffie-Hellman group:#2 (1024 bit)
    lifetime:5000 seconds, no volume limit
Protection suite priority 20
    encryption algorithm:DES - Data Encryption Standard (56 bit keys)
    hash algorithm:Secure Hash Standard
    authentication method:preshared Key
    Diffie-Hellman group:#1 (768 bit)
    lifetime:10000 seconds, no volume limit
Default protection suite
    encryption algorithm:DES - Data Encryption Standard (56 bit keys)
    hash algorithm:Secure Hash Standard
    authentication method:Rivest-Shamir-Adelman Signature
    Diffie-Hellman group:#1 (768 bit)
    lifetime:86400 seconds, no volume limit
```

Note Although the output shows “no volume limit” for the lifetimes, you can configure only a time lifetime (such as 86,400 seconds); volume-limit lifetimes are not configurable.

Configuring a service-ipsec Interface with a Dynamic Profile: Example

The following shows how to configure a service-ipsec interface with a dynamic profile:

```
ipv4 access-list acl1
 10 permit ipv4 any any
!
interface service-ipsec1
 ipv4 address 44.44.44.44 255.255.255.0
 profile ipsec-profile1
 tunnel source 100.0.0.1
 service-location preferred-active 0/4/0
!

crypto isakmp
crypto isakmp policy 10
 authentication pre-share
```

```
 group 5
 encryption 3des
 lifetime 86400
!
crypto keyring ring1 vrf default
 pre-shared-key address 40.0.0.1 255.255.255.255 key key1
!
crypto isakmp profile ike-profile1
 keyring ring1
 match identity address 40.0.0.0/16 vrf default
  set interface service-ipsec1
 !
!
crypto isakmp keepalive 60 5
crypto ipsec transform-set tsfm1 esp-3des esp-md5-hmac
!
crypto ipsec profile ipsec-profile1
 set type dynamic
 match acl1 transform-set tsfm1
!
```

Configuring Easy VPN with a Local AAA: Example

The following example shows how to configure Easy VPN with a local AAA:

```
aaa authorization network author-net-local local
aaa authentication login authen-net-local local
aaa authentication login author-net-local local

local pool
   ipv4 pool-1 20.20.20.4 20.20.20.255
!
ipv4 access-list acl-3
    10 permit ipv4 any any
!
interface Loopback30
    ipv4 address 10.20.100.1 255.255.255.255
interface Loopback31
    ipv4 address 2.1.0.5 255.255.255.255
!
interface Loopback33
    ipv4 address 10.20.100.3 255.255.255.255
interface MgmtEth0/0/CPU0/0
    ipv4 address 3.1.73.1 255.255.0.0
!
interface GigabitEthernet0/1/0/0.1
    ipv4 address 10.0.83.2 255.255.255.0
    dot1q vlan 83
!
interface GigabitEthernet0/1/0/0.2
    ipv4 address 10.0.81.4 255.255.255.0
    dot1q vlan 81
!
interface GigabitEthernet0/1/0/1
    ipv4 address 2.0.0.1 255.0.0.0
    negotiation auto
!
interface service-ipsec3
    ipv4 address 30.3.3.3 255.255.0.0
    profile ipsec-prof-ezvpn
    tunnel source 10.20.100.3
```

```
    service-location preferred-active 0/2/0
!
crypto isakmp client configuration group group-a
    key group-a-key
    pool pool-1
!
crypto isakmp
crypto isakmp policy 30
    authentication pre-share
    group 2
    encryption aes
    lifetime 180
!
crypto isakmp profile isakmp-prof3
    client authentication list authen-net-local
    match identity group group-a
        set interface service-ipsec3
!
isakmp authorization list author-net-local
!
crypto ipsec transform-set tsfm3
    transform esp-3des esp-sha-hmac
!
crypto ipsec profile ipsec-prof-ezvpn
    set type dynamic
    match acl-3 transform-set tsfm3
!
```

Configuring VRF-Aware: Example

The following example shows how to configure VRF-aware:

```
ipv4 access-list acl-2_5-1
    10 permit ipv4 any any
ipv4 access-list acl-2_5-4
    10 permit ipv4 host 2.6.1.3 host 1.7.1.3
vrf IVRF1
!
vrf IVRF2
!
vrf IVRF3
!
vrf FVRF
!
interface GigabitEthernet0/1/0/0.1
    vrf FVRF
    ipv4 address 10.0.83.2 255.255.255.0
!
interface GigabitEthernet0/1/0/1.1
    vrf IVRF1
    ipv4 address 2.6.0.1 255.255.0.0
    dot1q vlan 61
!
interface GigabitEthernet0/1/0/1.2
    vrf IVRF2
    ipv4 address 2.6.1.1 255.255.0.0
    dot1q vlan 62
!
interface GigabitEthernet0/1/0/1.3
    vrf IVRF3
    ipv4 address 2.6.0.1 255.255.0.0
```

```
    dot1q vlan 63
!
interface GigabitEthernet0/1/0/0.11
    vrf FVRF
    ipv4 address 10.0.91.1 255.255.255.0
    dot1q vlan 91
!
interface GigabitEthernet0/1/0/0.12
    vrf FVRF
    ipv4 address 10.0.92.1 255.255.255.0
    dot1q vlan 92
!
interface GigabitEthernet0/1/0/0.13
    vrf FVRF
    ipv4 address 10.0.93.1 255.255.255.0
    dot1q vlan 93
interface service-ipsec15
    vrf IVRF1
    ipv4 address 115.115.115.115 255.255.0.0
    service-location preferred-active 0/2/0
    profile ipsec-prof15 tunnel vrf FVRF
    tunnel source 10.20.100.15
interface service-ipsec16
    vrf IVRF2
    ipv4 address 116.116.116.116 255.255.0.0
    service-location preferred-active 0/2/0
    profile ipsec-prof16 tunnel vrf FVRF
    tunnel source 10.20.100.16
    tunnel destination 10.0.85.2

router static
    address-family ipv4 unicast
        1.7.0.3/32 service-ipsec15
        1.7.0.4/32 service-ipsec15  vrf FVRF
    address-family ipv4 unicast
    10.0.0.0/16 10.0.83.1

crypto isakmp
crypto isakmp policy 60
    authentication pre-share
    hash sha
    group 5
    encryption aes
    lifetime 86400
!
crypto keyring kr11 vrf FVRF
    pre-shared-key address 10.0.91.2 255.255.255.255 key key-vrf
    pre-shared-key address 10.0.92.2 255.255.255.255 key key-vrf
    pre-shared-key address 10.0.93.2 255.255.255.255 key key-vrf
!
crypto keyring kr12 vrf FVRF
    local-address 10.20.100.16
    pre-shared-key address 0.0.0.0 0.0.0.0 key key16
!
crypto isakmp profile isakmp-prof6
    keyring kr11
    match identity address 10.0.91.2/32 vrf FVRF
        set interface service-ipsec15
    match identity address 10.0.92.2/32 vrf FVRF
        set interface service-ipsec15
    match identity address 10.0.93.2/32 vrf FVRF
        set interface service-ipsec15
!
!
```

```
crypto isakmp profile isakmp-prof7
    keyring kr12
    match identity address 10.0.85.2/32 vrf FVRF
        set interface service-ipsec16
!
!
crypto ipsec transform-set tsfm5
    transform ah-sha-hmac esp-aes
crypto ipsec transform-set tsfm15
    transform esp-3des esp-md5-hmac
crypto ipsec transform-set tsfm16
    transform esp-aes esp-sha-hmac

crypto ipsec profile ipsec-prof15
    set type dynamic
    match acl-2_5-1 transform-set tsfm15
!
crypto ipsec profile ipsec-prof16
    match acl-2_5-4 transform-set tsfm16
!
```

Additional References

The following sections provide references related to implementing the IKE security protocol.

Related Documents

Related Topic	Document Title
IKE security protocol commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Internet Key Exchange Security Protocol Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5
IPSec conceptual information and configuration tasks	*Implementing IPSec Network Security on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*, Release 3.5

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
RFC 2409	*The Internet Key Exchange*
RFC 2408	*Internet Security Association and Key Management Protocol (ISAKMP) Internet Draft*

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

Implementing Keychain Management on Cisco IOS XR Software

Keychain management is a common method of authentication to configure shared secrets on all the entities, which 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.

Feature History for Implementing Keychain Management on Cisco IOS XR Software

Release	Modification
Release 3.3.0	This feature was introduced on Cisco CRS-1 and Cisco XR 12000 Series Router.
Release 3.4.0	• Support for the MAC authentication algorithm was added. • Support for hitless key rollover and key acceptance tolerance were added.
Release 3.5.0	Support for hitless key rollover for Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) was added.

Contents

- Restrictions for Implementing Keychain Management, page SC-75
- Information About Implementing Keychain Management, page SC-76
- How to Implement Keychain Management, page SC-76
- Configuration Examples for Implementing Keychain Management, page SC-87
- Additional References, page SC-88

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. For information about BGP, OSPF, and IS-IS keychain configurations, see *Cisco IOS XR Routing Configuration Guide*.

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.

To implement keychain management, you must understand the following concepts:

- Lifetime of a Key, page SC-76

Lifetime of a 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:

- Configuring a Keychain, page SC-77 (required)
- Configuring a Tolerance Specification to Accept Keys, page SC-78 (required)
- Configuring a Key Identifier for the Keychain, page SC-79 (required)
- Configuring the Text for the Key String, page SC-81 (required)

- Determining the Valid Keys, page SC-82 (optional)
- Configuring the Keys to Generate Authentication Digest for the Outbound Application Traffic, page SC-84 (required)
- Configuring the Cryptographic Algorithm, page SC-85 (required)

Configuring a Keychain

This task configures a name for the keychain.

You can create or modify the name of the keychain.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **end**
   or
   commit
4. **show key chain** *key-chain-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**key chain** *key-chain-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# key chain isis-keys`
	`RP/0/RP0/CPU0:router(config-isis-keys)#`
		Creates a name for the keychain.
		
		Note Configuring only the keychain name without any key identifiers is considered a nonoperation. When you exit the configuration, the router does not prompt you to commit changes until you have configured the key identifier and at least one of the global configuration mode attributes or keychain-key configuration mode attributes (for example, lifetime or key string).

	Command or Action	Purpose

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 4	**show key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router# show key chain isis-keys`	(Optional) Displays the name of the keychain. **Note** The *key-chain-name* argument is optional. If you do not specify a name for the *key-chain-name* argument, all the keychains are displayed.

What to Do Next

After completing keychain configuration, see the Configuring a Tolerance Specification to Accept Keys section.

Configuring a 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.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **accept-tolerance** [*value* | **infinite**]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.
Step 3	**accept-tolerance** [*value* | **infinite**] **Example:** `RP/0/RP0/CPU0:router(config-isis-keys)# accept-tolerance infinite`	Configures a tolerance value to accept keys for the keychain. • 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 the tolerance specification is infinite.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Key Identifier for the Keychain

This task configures a key identifier for the keychain.

You can create or modify the key for the keychain.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.
Step 3	**key** *key-id* **Example:** `RP/0/RP0/CPU0:router(config-isis-keys)# key 8`	Creates a key for the keychain. The key ID number is translated from decimal to hexadecimal to create the command mode subprompt. • Use the *key-id* argument as a 48-bit integer.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After configuring a key identifier for the keychain, see the Configuring the Text for the Key String section.

Configuring the Text for the Key String

This task configures the text for the key string.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **key-string** [**clear** | **password**] *key-string-text*
5. **end**

   or

   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.

Step 3	**key** *key-id*
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

	Command or Action	Purpose
Step 4	**key-string** [**clear** | **password**] *key-string-text* **Example:** `RP/0/RP0/CPU0:router(config-isis-keys-0x8)# key-string password 8`	Specifies the text string for the key. • Use the **clear** keyword to specify the key string in clear text form; use the **password** keyword to specify the key in encrypted form.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After configuring the text for the key string, see the Configuring the Keys to Generate Authentication Digest for the Outbound Application Traffic section.

Determining the Valid Keys

This task determines the valid keys for local applications to authenticate the remote peers.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **accept-lifetime** *start-time* [**duration** *durationvalue* | **infinite** | *end-time*]
5. **end**

   or

   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a a name for the keychain.

Step 3	**key** *key-id*
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

Step 4	**accept-lifetime** *start-time* [**duration** *durationvalue* | **infinite** | *end-time*]
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
	`accept-lifetime 1:00:00 october 24 2005 infinite`
		(Optional) Specifies the validity of the key lifetime in terms of clock time.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
	`commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Keys to Generate Authentication Digest for the Outbound Application Traffic

This task configures the keys to generate authentication digest for the outbound application traffic.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **send-lifetime** *start-time* [**duration** *durationvalue* | **infinite** | *end-time*]
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.

Step 3	**key** *key-id*
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

	Command or Action	Purpose

Step 4	**send-lifetime** *start-time* [**duration** *durationvalue* | **infinite** | *end-time*]
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# send-lifetime 1:00:00 october 24 2005 infinite`
		(Optional) Specifies the set time period during which an authentication key on a keychain is valid to be sent. You can specify the validity of the key lifetime in terms of clock time.
		In addition, you can specify a start-time value and one of the following values:
		• **duration** keyword (seconds)
		• **infinite** keyword
		• *end-time* argument
		If you intend to set lifetimes on keys, Network Time Protocol (NTP) or some other time synchronization method is recommended.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Cryptographic Algorithm

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

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*

4. **cryptographic-algorithm** [**HMAC-MD5 | HMAC-SHA1-12 | HMAC-SHA1-20 | MD5 | SHA-1**]
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**key chain** *key-chain-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# key chain isis-keys`
	`RP/0/RP0/CPU0:router(config-isis-keys)#`
		Creates a name for the keychain.

Step 3	**key** *key-id*
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

	Command or Action	Purpose
Step 4	**cryptographic-algorithm** [**HMAC-MD5** | **HMAC-SHA1-12** | **HMAC-SHA1-20** | **MD5** | **SHA-1**] **Example:** `RP/0/RP0/CPU0:router(config-isis-keys-0x8)# cryptographic-algorithm MD5`	Specifies the choice of the cryptographic algorithm. You can choose from the following list of algorithms: • HMAC-MD5 • HMAC-SHA1-12 • HMAC-SHA1-20 • MD5 • SHA-1

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing Keychain Management

This section provides the following configuration example:

- Configuring Keychain Management: Example, page SC-87

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
end

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

show key chain isis-keys

Key-chain: isis-keys/ -

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

Additional References

The following sections provide references related to implementing keychain management.

Related Documents

Related Topic	Document Title
Keychain management commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Keychain Management Commands on Cisco IOS XR Software* module in *Cisco IOS XR System Security Command Reference*, Release 3.5

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

Implementing IPSec Network Security on Cisco IOS XR Software

IP Security (IPSec) provides security for transmission of sensitive information over unprotected networks such as the Internet. IPSec acts at the network layer, protecting and authenticating IP packets between participating IPSec devices (“peers”), such as Cisco routers.

With IPSec, data can be sent across a public network without observation, modification, or spoofing, which enables applications, such as Virtual Private Networks (VPNs), including intranets, extranets, and remote user access.

IPSec for Cisco IOS XR supports the following two types of traffic:

- IPSec for locally sourced traffic or traffic terminated on the router. This mode is supported on both Cisco CRS-1 and Cisco XR 12000 Series Router. Either tunnel-ipsec interfaces or a transport entity are used. This type is also called *software-based IPSec*.
- IPSec for transit traffic is supported on the Cisco XR 12000 Series Router IPSec VPN SPA. This mode is also called *hardware-based IPSec*. Both service-ipsec and service-gre interfaces are used for this type.

This module describes the tasks that you need to implement IPSec network security on your Cisco IOS XR network.

Note For a complete description of the IPSec network security commands used in this chapter, see the *IPSec Network Security Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference* publication. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing IPSec Network Security on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.3.0	No modification.

Release 3.4.0	• Support was added for Cisco XR 12000 Series Router IPSec VPN SPA. • The **crypto ipsec chkpt-disabled** command was removed; therefore, the Configuring Checkpointing section was removed.
Release 3.5.0	• The Multiprotocol Label Switching (MPLS) Encapsulated Packets on Inbound Direction feature was added. • IPSec—SNMP support feature was added on the Cisco XR 12000 Series Router IPSec VPN SPA.

Contents

- Prerequisites for Implementing IPSec Network Security, page SC-92
- Restrictions for Implementing IPSec Network Security, page SC-93
- Restrictions for Implementing IPSec Network with a Cisco IPSec VPN SPA, page SC-93
- Information About Implementing IPSec Networks, page SC-94
- Information About an IPSec Network with a Cisco IPSec VPN SPA on Cisco IOS XR Software, page SC-101
- How to Implement General IPSec Configurations for IPSec Networks, page SC-104
- How to Implement IPSec Network Security for Locally Sourced and Destined Traffic, page SC-129
- How to Implement IPSec Network Security for VPNs, page SC-132
- Configuration Examples for Implementing IPSec Network Security for Locally Sourced Traffic and Destined Traffic, page SC-140
- Configuration Examples for an IPSec Network with a Cisco IPSec VPN SPA, page SC-142
- Additional References, page SC-147

Prerequisites for Implementing IPSec Network Security

The following prerequisites are required to implement IPSec network security:

- You must be in a user group associated with a task group that includes the proper task IDs for security commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- You must install and activate the Package Installation Envelope (PIE) for the security software.

  For detailed information about optional PIE installation, see the *Cisco IOS XR System Management Configuration Guide*.

Note For Cisco XR 12000 Series Router IPSec VPN SPA, you must install a service software PIE.

- You must configure Internet Key Exchange (IKE), as described in the *Implementing Internet Key Exchange Security Protocol on Cisco IOS XR Software* module.

Restrictions for Implementing IPSec Network Security

If you use Network Address Translation (NAT), you should configure static NAT translations so that IPSec will work properly. In general, NAT translation should occur before the router performs IPSec encapsulation; in other words, IPSec should be working with global addresses.

Note You should be using static crypto profiles.

Restrictions for Implementing IPSec Network with a Cisco IPSec VPN SPA

The following restrictions are known to implement IPSec network with a Cisco XR 12000 Series Router IPSec VPN SPA:

- Clear GRE is not supported. Only secure generic routing encapsulation (GRE) is supported by the Cisco XR 12000 Series Router IPSec VPN SPA. To configure the Cisco XR 12000 Series Router IPSec VPN SPA, you can use either service-ipsec or service-gre interfaces.
- Dynamic virtual interfaces are not supported.
- Dynamic Multipoint VPN (DMVPN) is not supported.
- Multicast is supported on interfaces in global VRF.

The following restrictions are known when implementing IPSec with the Cisco XR 12000 Series Router IPSec VPN SPA for Internet Security Association Key Management Protocol (ISAKMP) and IPSec profile configurations:

- One IPSec profile is configured on a virtual interface (for example, service ipsec or service-gre). A profile has one or more access control lists (ACLs). Each ACL has one or more access control entry (ACE). ACLs and ACEs cannot intersect.
- With both tunnel source and tunnel destination defined, a dynamic profile configuration on a virtual interface is rejected.
- Multiple virtual interfaces are attached to ISAKMP profiles; however, a virtual interface is referenced from a single ISAKMP profile only. The constraint is required to uniquely identify the virtual interface, ISAKMP profile, and IPSec profile as an IKE initiator and IKE responder.

The following restrictions are known to implement Cisco XR 12000 Series Router IPSec VPN SPA for the tunnel source address:

- Virtual interfaces that use the same tunnel source and FVRF are configured on the same Cisco XR 12000 Series Router IPSec VPN SPA.
- When NAT traversal is active for two tunnels that share the same source and destination address and FVRF under two different virtual interfaces, IP packets that require fragmentation are dropped.

Information About Implementing IPSec Networks

To implement IP network security, you should understand the following concepts:

- Crypto Profiles, page SC-94
- Dynamic Crypto Profiles, page SC-95
- Crypto Access Lists, page SC-95
- Transform Sets, page SC-96
- Global Lifetimes for IPSec Security Associations, page SC-96
- Checkpointing, page SC-98
- DF Bit Override Functionality with IPSec Tunnels, page SC-98
- IPSec Antireplay Window, page SC-98
- IPSec NAT Transparency, page SC-99
- IPSec Security Association Idle Timers, page SC-99
- Prefragmentation for Cisco IPSec VPN SPAs, page SC-99
- Reverse-Route Injection, page SC-100
- IPSec—SNMP Support, page SC-101

Note For information about IPSec Quality of Service (QoS), refer to the *Cisco IOS XR Quality of Service Configuration Guide*.

Crypto Profiles

Crypto profile entries created for IPSec combine the various parts used to set up IPSec security associations (SAs), including the following:

- Traffic that should be protected by IPSec (per a crypto access list)
- Granularity of the flow to be protected by a set of SAs
- IPSec security that should be applied to this traffic (selecting from a list of one or more transform sets)
- Other parameters that might be necessary to define an IPSec SA

Crypto profiles are applied to IPSec interfaces (for example, tunnel-ipsec, service-ipsec, and service) or crypto transport.

If the access control lists (ACLs) specified within the profile match any outbound IP traffic, the IP traffic is protected by IPSec. The SA is established with the remote peer by IKE.

When using service-gre interfaces, the profile, which is attached to the interface, is not configured with an explicit ACL. Instead, all traffic, which is destined to the GRE tunnel, is protected by IPSec.

The policy described in the crypto profile entries is used during the negotiation of SAs. If the local router initiates the negotiation, it uses the policy specified in the static crypto profile entries to create the offer to be sent to the specified IPSec peer. If the IPSec peer initiates the negotiation, the local router checks the policy associated with the interface or profile associated with the identity specified in the ISAKMP profile, which is being used to decide whether to accept or reject the peer's request (offer).

For IPSec to succeed between two IPSec peers, both peers' crypto profile entries must contain compatible configuration statements. When two peers try to establish an SA, each must have at least one crypto profile entry that is compatible with one of the other peer's crypto profile entries. For two crypto profile entries to be compatible, they must at least meet the following criteria:

- The crypto profile entries must contain compatible crypto access lists. In the case where the responding peer is using dynamic crypto profiles, the entries in the local crypto access list must be "permitted" by the peer's crypto access list.
- The crypto profile entries must have at least one transform set in common.

Note

- Crypto profiles cannot be shared, that is, the same profile cannot be attached to multiple tunnel-IPSec interfaces or an interface and transport mode IPSec.
- The restriction is only for ipsec-tunnel interface or transport and not service-ipsec or service-gre interfaces.

Dynamic Crypto Profiles

A dynamic crypto profile entry is essentially a crypto profile entry without all the parameters configured. It acts as a policy template in which the missing parameters are later dynamically configured (as the result of an IPSec negotiation) to match a remote peer's requirements. This allows remote peers to exchange IPSec traffic with the router even if the router does not have a crypto profile entry specifically configured to meet all of the remote peer's requirements.

Dynamic crypto profiles are not used by the router to initiate new IPSec SAs with remote peers. Dynamic crypto profiles are used when a remote peer tries to initiate an IPSec SA with the router. Dynamic crypto profiles are also used in evaluating traffic.

If the router accepts the peer's request, at the point that it installs the new IPSec SAs it implicitly installs a temporary crypto profile entry. This entry is filled in with the results of the negotiation. At this point, the router performs normal processing, using this temporary crypto profile entry as a normal entry, even requesting new SAs if the current ones are expiring (based upon the policy specified in the temporary crypto profile entry). After the flow expires (that is, all of the corresponding SAs expire), the temporary crypto profile entry is then removed.

For static crypto profile entries, if outbound traffic matches a **permit** statement in an access list and the corresponding SA is not yet established, the router will initiate new SAs with the remote peer. In the case of dynamic crypto profile entries, if no SA existed, the traffic would be dropped (because dynamic crypto profiles are not used for initiating new SAs).

Crypto Access Lists

Crypto access lists are used to define the IP traffic that is and is not protected by crypto. For example, access lists can be created to protect all IP traffic between Subnet A and Subnet Y or Telnet traffic between Host A and Host B.

The access lists themselves are not specific to IPSec. It is the crypto profile entry referencing the specific access list that defines whether IPSec processing is applied to the traffic matching a **permit** in the access list.

Crypto access lists associated with IPSec crypto profile entries have four primary functions:

- Select outbound traffic to be protected by IPSec (permit = protect).
- Indicate the data flow to be protected by the new SAs (specified by a single **permit** entry) when initiating negotiations for IPSec SAs.
- Process inbound traffic to filter and discard traffic that should have been protected by IPSec.
- Determine whether to accept requests for IPSec SAs on behalf of the requested data flows when processing IKE negotiation from the IPSec peer. (Negotiation is done only for **ipsec-isakmp** crypto profile entries.) To be accepted, if the peer initiates the IPSec negotiation, it must specify a data flow that is "permitted" by a crypto access list associated with an **ipsec-isakmp** crypto profile entry.

If you want certain traffic to receive one combination of IPSec protection (for example, authentication only) and other traffic to receive a different combination of IPSec protection (for example, both authentication and encryption), you need to create two different crypto access lists to define the two different types of traffic.

Transform Sets

A transform set represents a certain combination of security protocols and algorithms. During the IPSec SA negotiation, the peers agree to use a particular transform set for protecting a particular data flow.

You can specify multiple transform sets and then one or more of these transform sets in a crypto profile entry. The transform set defined in the crypto profile entry is used in the IPSec SA negotiation to protect the data flows specified by that crypto profile entry's access list.

During IPSec SA negotiations with IKE, the peers search for a transform set that is the same at both peers. When such a transform set is found, it is selected and applied to the protected traffic as part of both peers' IPSec SAs.

If you change a transform set definition, the change is applied only to crypto profile entries that reference the transform set. The change will not be applied to existing SAs, but is used in subsequent negotiations to establish new SAs. If you want the new settings to take effect sooner, you can clear all or part of the SA database by using the **clear crypto ipsec sa** command.

Global Lifetimes for IPSec Security Associations

You can change the global lifetime values that are used when negotiating new IPSec SAs.

Two lifetimes exist: a "timed" lifetime and "traffic-volume" lifetime. An SA expires after the first of these lifetimes is reached. The default lifetimes are 3600 seconds (1 hour) and 4,194,303 kilobytes (10 MBps for 1 hour).

In addition, a lifetime per profile is supported. If a profile is configured with a lifetime, it is overriding the global definition.

If you change a global lifetime, the new lifetime value is not applied to currently existing SAs, but is used in the negotiation of subsequently established SAs. If you want to use the new values immediately, you can clear all or part of the SA database. See the **clear crypto ipsec sa** command for more details.

IPSec SAs use one or more shared secret keys. These keys and their SAs time out together.

Assuming that the particular crypto profile entry does not have lifetime values configured, when the router requests new SAs it specifies its global lifetime values in the request to the peer; it uses this value as the lifetime of the new SAs. When the router receives a negotiation request from the peer, it uses the smaller of either the lifetime value proposed by the peer or the locally configured lifetime value as the lifetime of the new SAs.

The SA (and corresponding keys) expire according to whichever comes sooner, either after the number of seconds has passed (specified by the **seconds** keyword) or amount of traffic in kilobytes is passed (specified by the **kilobytes** keyword).

A new SA is negotiated *before* the lifetime threshold of the existing SA is reached, to ensure that a new SA is ready for use when the old one expires. The new SA is negotiated approximately 30 seconds before the **seconds** lifetime expires or when the volume of traffic through the tunnel reaches 300 kilobytes less than the **kilobytes** lifetime (whichever comes first).

If no traffic has passed through the tunnel during the entire life of the SA, a new SA is not negotiated when the lifetime expires. Instead, a new SA is negotiated only when IPSec sees another packet that should be protected.

Manual IPSec Security Associations

The use of manual security associations is a result of a prior arrangement between the users of the local router and the IPSec peer. The two parties can begin with manual security associations, but must move to using security associations established through IKE, or the remote party's system cannot support IKE. If IKE is not used to establish security associations, there is no negotiation of security associations, so the configuration information in both systems must be the same for traffic to be processed successfully by IPSec.

The local router can simultaneously support manual and IKE-established security associations, even within a single crypto profile entry. There is very little reason to disable IKE on the local router (unless the router supports only manual security associations, which is unlikely).

Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) ensures that a given key of an IPSec security association is not derived from any other secret, such as some other keys. In other words, if someone broke a key, PFS ensures that the attacker is not able to derive any other key. If PFS is not enabled, someone can hypothetically break the IKE SA secret key, copy all the IPSec-protected data, and use knowledge of the IKE SA secret to compromise the IPSec SAs set up by this IKE SA. With PFS, breaking IKE does not give an attacker immediate access to IPSec. The attacker needs to break each IPSec SA individually.

During negotiation, the **set pfs** command causes IPSec to request PFS when requesting new security associations for the crypto profile entry. If the **set pfs** command statement does not specify a group, the default (group1) is sent. If the peer initiates the negotiation and the local configuration specifies PFS, the remote peer must perform a PFS exchange or the negotiation fails. If the local configuration does not specify a group, a default of group1 is assumed, and an offer of either group1, group2, or group5 is accepted. If the local configuration specifies group2 or group5, the group must be part of the offer from the peer or the negotiation fails. If the local configuration does not specify PFS, the configuration accepts any offer of PFS from the peer.

Checkpointing

IPSec checkpoints SAs in the local database. If an IPSec process restarts, SAs are retrieved from the local database and need not be re-established with remote peers.

DF Bit Override Functionality with IPSec Tunnels

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

A *Don't Fragment (DF) bit* is a bit within the IP header that determines whether a router is allowed to fragment a packet. The DF Bit Override Functionality with IPSec Tunnels feature allows you to specify whether your router can clear, set, or copy the DF bit from the encapsulated header.

Some configurations have hosts that perform the following functions:

- Set the DF bit in packets they send.
- Use firewalls that block Internet Control Message Protocol (ICMP) errors from outside the firewall, preventing hosts from learning about the maximum transmission unit (MTU) size outside the firewall.
- Use IPSec to encapsulate packets to reduce the available MTU size.

If your configurations have hosts that prevent them from learning about the available MTU size, you can configure your router to clear the DF bit and fragment the packet.

The DF Bit Override Functionality with IPSec Tunnels feature allows you to configure the setting of the DF bit when encapsulating IPSec tunnels for IPSec traffic on a global or per-interface level. Thus, if the DF bit is set to clear, routers can fragment packets regardless of the original DF bit setting.

IPSec Antireplay Window

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

Cisco IPSec authentication provides antireplay protection against an attacker duplicating encrypted packets, by assigning a unique sequence number to each encrypted packet. (Security association [SA] antireplay is a security service in which the receiver can reject old or duplicate packets to protect itself against replay attacks.) The decryptor checks off the sequence numbers that it has seen before. The encryptor assigns sequence numbers in an increasing order. The decryptor remembers the value X of the highest sequence number that it has already seen. N is the window size, and the decryptor also remembers whether it has seen packets having sequence numbers from X-N+1 through X. Any packet with the sequence number smaller than X-N is discarded. Currently, N is set at 64, so only 64 packets can be kept in the memory of the decryptor.

At times, however, the 64-packet window size is not sufficient. For example, Cisco quality of service (QoS) gives priority to high-priority packets, which could cause some low-priority packets to be discarded even though they could be one of the last 64 packets received by the decryptor. The IPSec Antireplay Window: Expanding and Disabling feature allows you to expand the window size, allowing the decryptor to keep more than 64 packets in its memory.

IPSec NAT Transparency

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

Previously, a standard IPSec Virtual Private Network (VPN) tunnel does not work if there were one or more Network Address Translator (NAT) or Point Address Translation (PAT) points in the delivery path of the IPSec packet. The IPSec NAT transparency feature makes NAT IPSec-aware; therefore, allowing remote access users to build IPSec tunnels to home gateways.

IPSec Security Association Idle Timers

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

When a router running Cisco IOS XR software creates an IPSec SA for a peer, resources must be allocated to maintain the SA. The SA requires both memory and several managed timers. For idle peers, these resources are wasted. If enough resources are wasted by idle peers, the counter could be prevented from creating new SAs with other peers. The IPSec security feature introduces a configurable idle timer to monitor SAs for activity, allowing SAs for idle peers to be deleted. The idle timers are configured either globally or on a crypto profile basis.

Prefragmentation for Cisco IPSec VPN SPAs

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

When a packet is nearly the size of the maximum transmission unit (MTU) of the outbound link of the encrypting router and it is encapsulated with IPSec headers, the packet is likely to exceed the MTU of the outbound link. The packet causes packet fragmentation after encryption, which makes the decrypting router reassemble in the process path. Prefragmentation for Cisco IPSec VPN SPAs increases the decrypting router's performance by enabling it to operate in the high-performance CEF path instead of the process path.

This feature allows an encrypting router to predetermine the encapsulated packet size from information available in transform sets, which are configured as part of the IPSec SA. If it is predetermined that the packet exceeds the MTU of the output interface, the packet is fragmented before encryption. This function avoids process-level reassembly before decryption and helps improve decryption performance and overall IPSec traffic throughput.

Prefragmentation for the Cisco IPSec VPN SPA functionality depends on the service-ipsec interface from the **crypto ipsec df-bit** command configuration and the incoming packet “do not fragment” (DF) bit state (see Table 4).

Table 4 Pre-Fragmentation for Cisco IPSec VPN SPA Dependencies

Pre-Fragmentation for IPSec VPN SPAs Feature State (Enabled or Disabled)	Service IPSec Interface "crypto ipsec df-bit" Configuration	Incoming Packet DF Bit State	Result
Enabled	crypto ipsec df-bit copy	0	Fragmentation occurs before encryption.
Enabled	crypto ipsec df-bit copy	1	Packets are dropped.
Enabled	crypto ipsec df-bit clear	0	Fragmentation occurs before encryption.
Enabled	crypto ipsec df-bit clear	1	Packets are sent to egress interfaces (not fragmented before encryption).[1]
Enabled	crypto ipsec df-bit set	0	Fragmentation occurs before encryption.
Enabled	crypto ipsec df-bit set	1	Packets are dropped.
Disabled	crypto ipsec df-bit copy	0	Packets are sent to egress interfaces.[1]
Disabled	crypto ipsec df-bit copy	1	Packets are dropped.
Disabled	crypto ipsec df-bit clear	0	Packets are sent to egress interfaces.[1]
Disabled	crypto ipsec df-bit clear	1	Packets are sent to egress interfaces.[1]
Disabled	crypto ipsec df-bit set	0	Packets are sent to egress interfaces.[1]
Disabled	crypto ipsec df-bit set	1	Packets are dropped.

1. A packet that is sent to egress interfaces can get fragmented on an egress LC under the following conditions: Packet exceeds the MTU of the egress physical interface. The df-bit is not set on the outer IP header.

Reverse-Route Injection

Reverse-Route Injection (RRI) is the ability of static routes to be automatically inserted into the routing process for those networks and hosts protected by a remote tunnel endpoint. These protected hosts and networks are known as remote proxy identities.

Each route is created on the basis of the remote proxy network and mask, with the next hop to this network being the remote tunnel endpoint. By using the remote tunnel endpoint as the next hop, the traffic is forced through the crypto process to be encrypted.

After the static route is created on the VPN router, this information is propagated to upstream devices, allowing them to determine the appropriate VPN router in which to send returning traffic to maintain IPSec state flows. Being able to determine the appropriate VPN router is particularly useful when multiple VPN routers are used at a site to provide load balancing or automatic switchover, or when the remote VPN devices are not accessible through a default route. Routes are created in either the global routing table or the appropriate virtual route forwarding (VRF) table.

IPSec—SNMP Support

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

The IPSec SNMP support feature allows you to specify the desired size of a tunnel history table by using the Cisco IOS XR CLI. The history table archives attribute and statistic information about the tunnel. A tunnel history table does not accompany every failure table, because every failure does not correspond to a tunnel. Thus, supported setup failures are recorded in the failure table, but an associated history table is not recorded because a tunnel was never set up.

Information About an IPSec Network with a Cisco IPSec VPN SPA on Cisco IOS XR Software

To implement an IPSec network with a Cisco IPSec VPN SPA, you should understand the following concepts:

- Cisco IPSec VPN SPA Overview, page SC-101
- Displaying the SPA Hardware Type, page SC-101
- Information About Security for VPNs with IPSec, page SC-102

Cisco IPSec VPN SPA Overview

Cisco IOS XR Software supports security protocols such as Authentication Header (AH), Encapsulating Security Payload (ESP), and Internet Key Exchange (IKE). The resources consumed by these activities are significant and make it difficult to achieve line-rate transmission speeds over VPNs. Using the Cisco IPSec VPN SPA enables you to send all VPN traffic coming from or going to the Internet through the SPA hardware. The SPA supports all IPSec-related processing. Packets coming from the trusted LAN are encrypted and sent through the Internet. Packets that are received from the WAN routers pass through the Cisco IPSec VPN SPA for IPSec processing (for example, decryption and validation of the packet before the packet is sent onto the trusted LAN).

The following three SIP card types are supported on the Cisco XR 12000 Series Router:

- SIP-401
- SIP-501
- SIP-601

Displaying the SPA Hardware Type

To verify that the SPA hardware type is installed on the Cisco XR 12000 Series Router, use the **show diag** command. In addition, you can use the **show platform** command to verify SPA hardware information.

Table 5 lists the hardware description that appears in the **show diag** command output for a Cisco XR 12000 Series Router IPSec VPN SPA.

Table 5 SPA Hardware Description in show diag Command

SPA	Description in show diag Command
SPA-IPSEC-2G	SPA-IPSEC-2G (0x3d7)

The following sample output is from the **show diag** command on the Cisco XR 12000 Series Router IPSec VPN SPA installed in slot 1:

```
RP/0/0/CPU0:router# show diag

SLOT  1 (RP/LC 1): Cisco 12000 Series SPA Interface Processor-600
  MAIN: type 117,  800-26102-01 rev B0 dev 0
        HW config: 0x01    SW key: 00-00-00
  PCA:  73-9863-02 rev B0 ver 8
        HW version 1.0  S/N SAD092306A5
  MBUS: Embedded Agent
        Test hist: 0x00    RMA#: 00-00-00    RMA hist: 0x00
  DIAG: Test count: 0x00000000    Test results: 0x00000000
  FRU:  Linecard/Module: 12000-SIP-600
        Route Memory: MEM-LC5-1024=
        Packet Memory: MEM-LC5-PKT-512=
  L3 Engine: 5 - ISE OC192 (10 Gbps)
  MBUS Agent Software version 2.49 (RAM) (ROM version is 3.6)
  Using CAN Bus A
  ROM Monitor version 17.1
  Fabric Downloader version used 3.9 (ROM version is 3.9)
  Primary clock is CSC1
  Board State is IOS-XR RUN
  Insertion time: Thu Jun 29 23:23:48 2006 (1w0d ago)
  DRAM size: 1073741824 bytes
  FrFab SDRAM size: 268435456 bytes
  ToFab SDRAM size: 268435456 bytes
  0 crashes since restart/fault forgive

  SPA Information:
        subslot 0/1/0: SPA-IPSEC-2G (0x3d7), status is ok
```

Information About Security for VPNs with IPSec

To implement security for VPNs with IPSec, you should understand the following concepts:

- IPSec Virtual Interfaces, page SC-102
- IPSec Load Balancing and High Availability, page SC-103
- VRF-aware IPSec, page SC-104
- MPLS Encapsulated Packets on Inbound Direction, page SC-104
- Reverse-Route Injection, page SC-100

IPSec Virtual Interfaces

IPSec virtual interfaces simplify configuration of IPSec for protection of remote links, support multicast, and simplify network management and load balancing.

Generic Routing Encapsulation (GRE) is a tunneling protocol that can encapsulate a wide variety of protocol packet types inside IP tunnels, creating a virtual point-to-point link to routers at remote points over an IP network.

When you configure an IPSec virtual interface, you must associate the service entity by using the **service-location** command. The association is done statically. The **service-location** command is mandatory. A virtual interface configuration is not fully verified until the service location is specified.

The following IPSec virtual interfaces are supported:

- Static IPSec service virtual interface (SVI)—Use the **interface service-ipsec** command to create a static IPSec SVI.

  The tunnel endpoints are defined by the **tunnel source** command and **tunnel destination** command. The tunnel source can be shared between interfaces. Although the address is internal to the router, the address is used only as a tunnel source. Unlike other internal router IP addresses, the tunnel source is not used to serve routing protocols, or other applications that are terminated on the router. The sole purpose for a tunnel source is tunneling.

  If a dynamic IPSec profile is attached to a static IPSec SVI, tunnel destination should not be configured and is negotiated when a new tunnel is created. In this case, it is possible that multiple IPSec tunnels can terminate on the same IPSec SVI and contain a different tunnel destination.

  A virtual interface must be associated with an IPSec service SPA. The **service-location** command specifies both active and standby locations for the interface. All interfaces that share the same tunnel source and tunnel VRF (FVRF) must be associated to the same location. The only event in which virtual interfaces share the same source address, destination address, and FVRF, is when NAT traversal is in effect.

- Static IPSec-protected GRE virtual interface—Use the **interface service-gre** command to create a static IPSec-protected GRE interface. When GRE is used with IPSec, only transport mode is supported. Only one IPSec profile can be attached to a GRE interface in which case one IPSec SA is created. Only point-to-point GRE interfaces are supported.

IPSec Load Balancing and High Availability

Load balancing and high availability are the mechanism in which IPSec handling is distributed between Cisco IPSec VPN SPAs, and the traffic diversion policy is used automatic switchover is required. You are able to configure the preferred active SPA (primary SPA) and preferred standby SPA (secondary SPA). If the active location is not functional, the standby location takes the active role. When both active and standby locations are functional, in the case of a failure in the active SPA, the secondary SPA serves as the primary location and traffic is diverted to that SPA. In addition, you have the option of configuring the auto-revert.

When the active location is functional (for example, after a failure), traffic is diverted back to the active location.

When auto-revert is not configured, if there is a failure in location A, location B (which was configured as standby) takes over and location A becomes the standby.

The preferred-active and preferred-standby SPAs cannot reside on the same line card.

Reverse Route Injection (RRI) is designed to simplify network design for Virtual Private Networks (VPNs) in which there is a requirement for redundancy or load balancing. For more information about RRI, see "Reverse-Route Injection" section on page SC-100.

VRF-aware IPSec

Each IPSec tunnel is associated with two VRF domains. The outer encapsulated domain belongs to one VRF domain, which is called the *front door VRF (FVRF)*, while the inner, protected IP packet belongs to another domain called *inside VRF (IVRF)*. Therefore, the local endpoint of the IPSec tunnel belongs to the FVRF, while the source and destination addresses of the inside packet belong to the IVRF.

Clear IP traffic is forwarded from an internal VRF to a remote site or host within the VRF over IPSec tunnels. The IVRF is determined on the SVI by using the **vrf** command. The encrypted packets going over the IPSec tunnel are forwarded over the FVRF, which is configured on the SVI by using the **tunnel vrf** command. The tunnel source and destination are addresses of the FVRF. The encapsulated packets and the ACLs, which are configured in the IPSec profile, are all part of the IVRF.

MPLS Encapsulated Packets on Inbound Direction

The Multiprotocol Label Switching (MPLS) distribution protocol is a high-performance packet-forwarding technology that integrates the performance and traffic management capabilities of data link switching with the scalability, flexibility, and performance of network-layer routing.

The IPSec packet arrives from the Internet and is destined for the provider edge (PE) 2, which is also called the *IPSec terminator*. If the packet arrives at a PE1 outside of a VRF (for example, in the global table), the ingress PE1 pushes a label switched path (LSP) label onto the IPSec packet. The LSP packet is used to tunnel the IPSec packet to the egress PE, which is the IPSec terminator.

How to Implement General IPSec Configurations for IPSec Networks

This section contains the following procedures:

- Setting Global Lifetimes for IPSec Security Associations, page SC-105 (optional)
- Creating Crypto Access Lists, page SC-106 (required)
- Defining Transform Sets, page SC-108 (required)
- Configuring Crypto Profiles, page SC-109 (required)
- Applying Crypto Profiles to tunnel-ipsec Interfaces, page SC-130 (required)
- Applying Crypto Profiles to Crypto Transport, page SC-131 (required)
- Configuring the DF Bit for the Encapsulating Header in IPSec Tunnels, page SC-114
- Configuring the IPSec Antireplay Window: Expanding and Disabling, page SC-115
- Configuring IPSec NAT Transparency, page SC-118
- Configuring IPSec Security Association Idle Timers, page SC-120
- Disabling Prefragmentation for Cisco IPSec VPN SPAs, page SC-124
- Configuring Reverse-Route Injection in a Crypto Profile, page SC-127
- Configuring IPSec Failure History Table Size, page SC-128

Setting Global Lifetimes for IPSec Security Associations

This task sets global lifetimes for IPSec security associations.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*}
3. **end**
   or
   commit
4. **clear crypto ipsec sa** {*sa-id* | **all**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ipsec security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*} **Example:** `RP/0/RP0/CPU0:router(config)# crypto ipsec security-association lifetime seconds 2700`	Changes global lifetime values used when negotiating IPSec SAs. • The **seconds** *seconds* keyword and argument change the global "timed" lifetime for IPSec SAs. This form of the command causes the SA to time out after the specified number of seconds have passed. • The **kilobytes** *kilobytes* keyword and argument change the global "traffic-volume" lifetime for IPSec SAs. This form of the command causes the SA to time out after the specified amount of traffic (in kilobytes) has passed through the IPSec "tunnel" using the SA.

	Command or Action	Purpose

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 4	**clear cypto ipsec sa {*sa-id* | all}** **Example:** `RP/0/RP0/CPU0:router# clear crypto ipsec sa 100`	(Optional) Clears existing security associations, which causes any existing SAs to expire immediately. • Future SAs use the new lifetimes. • Any existing SAs expire according to the previously configured lifetimes. **Note** Using the **clear crypto ipsec sa** command with the **all** keyword clears the full SA database, which clears active security sessions. You may also specify the *sa-id* argument to clear an SA with a specific ID. For more information, see the **clear crypto ipsec sa** command.

Creating Crypto Access Lists

This task creates crypto access lists.

SUMMARY STEPS

1. **configure**
2. **ipv4 access-list** *name*
3. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**dscp** *dscp*] [**fragments**] [*packet-length operator packet-length value*] [**log | log-input**]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**ipv4 access-list** *name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# ipv4 access-list InternetFilter`
	`RP/0/RP0/CPU0:router(config-ipv4-acl)#`
		Specifies conditions to determine which IP packets are protected.
		• Enables or disables crypto for traffic that matches these conditions.
		• The **any** keyword should be used, as described in the "The any Keyword in Crypto Access Lists" section on page SC-129.
		• Follow with **permit** and **deny** statements, as appropriate. For more information, see "The any Keyword in Crypto Access Lists" section on page SC-129.

Step 3	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**dscp** *dscp*] [**fragments**] [*packet-length operator packet-length value*] [**log** | log-input] **Example:** `RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 100.0.1.0 0.0.0.255 30.0.1.0 0.0.0.255`	Sets a permit condition for an access list named Internetfilter.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Defining Transform Sets

This task defines a transform set.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec transform-set** *name*
   transform-set submode transform protocol
   transform-set submode **mode** {**transport** | **tunnel**}
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose

Step 2	**crypto ipsec transform-set** *name*
	transform-set submode **transform** *protocol*
	transform-set submode **mode** {**transport** | **tunnel**}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# crypto ipsec transform-set new`
	`RP/0/RP0/CPU0:router(config-transform-set new)# transform esp-sha-hmac`
		Defines a transform set.
		• Complex rules define which entries you can use for the transform arguments. These rules are explained in the command description for the **crypto ipsec transform-set** command.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-transform-set new)# end`
	or
	`RP/0/RP0/CPU0:router(config-transform-set new)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Crypto Profiles

This task configures static or dynamic crypto profiles.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec profile** *name*
3. **match** *acl-name* **transform-set** *transform-set-name*
4. **set pfs** {**group1** | **group2** | **group5**}
5. **set type** {**static** | **dynamic**}
6. **set transform-set** *transform-set-name*
7. **reverse-route**
8. **set security-association idle-time** *seconds*
9. **set security-association lifetime seconds** *seconds* **kilobytes** *kilobytes*
10. **set security-association replay disable**

11. **set session-key inbound ah** *spi hex-key-data*
12. **set session-key inbound esp** *spi* {**cipher** *hex-key-data* **authentication** *hex-key-data*}
13. **set session-key outbound ah** *spi hex-key-data*
14. **set session-key outbound esp** *spi* {**cipher** *hex-key-data* **authentication** *hex-key-data*}
15. **exit**
16. **end**
    or
    commit
17. **show crypto ipsec sa** [**sa-id** | **peer** *ip-address* | **profile** *profile-name* | **detail** | **fvrf** *fvrf-name* | **ivrf** *ivrf-name* | **location** *location*]
18. **show crypto ipsec summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ipsec profile** *name* **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec profile new`	Creates the IPSec profile and enters profile configuration mode.
Step 3	**match** *acl-name* **transform-set** *transform-set-name* **Example:** `RP/0/0/CPU0:router(config-new)# match sampleacl transform-set tset1`	Configures the ACL to use for packet classification, and if the packets need protecting, the transform set to use for IPSec processing. **Note** You can configure up to five different transform-sets. The **match transform-set** command is used in profiles that are attached to service-ipsec interfaces, tunnel-ipsec interfaces, and transport. The description for this command is similar to the **set transform-set** command but used on a different interface.
Step 4	**set pfs** {**group1** | **group2** | **group5**} **Example:** `RP/0/0/CPU0:router(config-new)# set pfs group5`	(Optional) Specifies that IPSec should ask for perfect forward secrecy (PFS) when requesting new security associations for this crypto profile entry, or should demand PFS in requests received from the IPSec peer.
Step 5	**set type {static | dynamic}** **Example:** `RP/0/0/CPU0:router(config-new)# set type dynamic`	(Optional) Sets the profile mode type. • Default is static mode, which means that the peer is identified in the configuration. • Dynamic mode lets the profile be dynamic, which means that SA negotiation from any authenticated peer is allowed.

	Command or Action	Purpose
Step 6	**set transform-set** *transform-set-name* **Example:** `RP/0/0/CPU0:router(config-new)# set transform-set ts1`	Specifies a list of transform sets in priority order. The **set transform-set** command is used in profiles that are attached to service-gre interfaces. The description for this command is similar to the **match transform-set** command but used on a different interface. **Note** You can configure up to five different transform-sets. Use the *transform-set-name* argument to name the transform-set. The maximum characters is 32.
Step 7	**reverse-route** **Example:** `RP/0/0/CPU0:router(config-new)# reverse-route`	Creates source proxy information for a crypto profile entry.
Step 8	**set security-association idle-time** *seconds* **Example:** `RP/0/0/CPU0:router(config-new)# set security-association idle-time 800`	Specifies the maximum time in which the current peer can be idle before the default peer is used. • Use the *seconds* argument to specify the number of seconds for which the current peer can be idle before the default peer is used. The valid values are 600 to 86400.

Step 9	**set security-association lifetime seconds** *seconds* **kilobytes** *kilobytes*
	
	Example:
	`RP/0/0/CPU0:router(config-new)# set security-association lifetime seconds 2700`
	`RP/0/0/CPU0:router(config-new)# set security-association lifetime kilobytes 2304000`
		Overrides (for a particular crypto profile entry) the global lifetime value, which is used when negotiating IP Security security associations.
		
		The example shows how to shorten lifetimes to reduce the risk that the keys could be compromised. The timed lifetime is shortened to 2700 seconds (45 minutes), and the traffic-volume lifetime is shortened to 2,304,000 KB (10 MBps for 30 minutes).
		
		• Use the **seconds** keyword to specify the number of seconds a security association lives before expiring. The range is from 120 to 86400.
		
		• Use the **kilobytes** keyword to specify the volume of traffic (in kilobytes) that can pass between IPSec peers using a given security association before that security association expires. The range is from 2560 to 536870912.

Step 10	**set security-association replay disable** **Example:** `RP/0/0/CPU0:router(config-new)# set security-association replay disable`	Disables replay checking for a particular crypto profile.

	Command or Action	Purpose
Step 11	**set session-key inbound ah** *spi hex-key-data* **Example:** `RP/0/0/CPU0:router(config-new)# set session-key inbound ah 300 11111111111111111111111111111111111111111111`	(Optional) Manually specifies the IP Security session keys to set the inbound IPSec session key for the Authentication Header (AH) protocol. The length of the keys should match the encryption or authentication method that is specified in the transform-set. • Use the *spi* argument to specify the security parameter index (SPI), a number that uniquely identifies a security association. The SPI is an arbitrary number you assign in the range of 256 to 4,294,967,295 (FFFF FFFF). • Use the *hex-key-data* argument to specify the session key; enter in hexadecimal format. This is an arbitrary hexadecimal string of 8, 16, or 20 bytes.
Step 12	**set session-key inbound esp** *spi* {**cipher** *hex-key-data* **authentication** *hex-key-data*} **Example:** `RP/0/0/CPU0:router(config-new)# set session-key inbound esp 300 cipher 0123456789012345 authentication 000011112222333344445555666677778888999 9`	(Optional) Manually specifies the IP Security session key to set the inbound IPSec session key for Encapsulation Security Protocol (ESP). The length of the keys should match the encryption or authentication method that is specified in the transform-set. • Use the *spi* argument to specify the SPI, a number that is used to uniquely identify a security association. The SPI is an arbitrary number you assign in the range of 256 to 4,294,967,295 (FFFF FFFF). • Use the **cipher** keyword to specify the key string to be used with the ESP encryption transform. • Use the *hex-key-data* argument to specify the session key; enter in hexadecimal format. This is an arbitrary hexadecimal string of 8, 16, or 20 bytes. • Use the **authentication** keyword to specify that the key string is used with the ESP authentication transform. The **authentication** keyword is required only when the transform set includes an ESP authentication transform.
Step 13	**set session-key outbound ah** *spi hex-key-data* **Example:** `RP/0/0/CPU0:router(config-new)# set session-key outbound ah 300 fedcbafedcbafedcbafedcbafedcbafedcbafedc`	(Optional) Manually specifies the IP Security session key to set the outbound IPSec session key for the AH protocol. The length of the keys should match the encryption or authentication method that is specified in the transform-set. • Use the *spi* argument to specify the security parameter index (SPI), a number that uniquely identifies a security association. The SPI is an arbitrary number you assign in the range of 256 to 4,294,967,295 (FFFF FFFF). • Use the *hex-key-data* argument to specify the session key; enter in hexadecimal format. This is an arbitrary hexadecimal string of 8, 16, or 20 bytes.

	Command or Action	Purpose
Step 14	**set session-key outbound esp** *spi* {**cipher** *hex-key-data* **authentication** *hex-key-data*} **Example:** `RP/0/0/CPU0:router(config-new)# set session-key outbound esp 300 cipher abcdefabcdefabcd authentication 99998888777766665555444433332222111110000`	(Optional) Manually specifies the IP Security session key to set the outbound IPSec session key for ESP. The length of the keys should match the encryption or authentication method that is specified in the transform-set. • Use the *spi* argument to specify the SPI, a number that is used to uniquely identify a security association. The SPI is an arbitrary number you assign in the range of 256 to 4,294,967,295 (FFFF FFFF). • Use the **cipher** keyword to specify the key string to be used with the ESP encryption transform. • Use the *hex-key-data* argument to specify the session key; enter in hexadecimal format. This is an arbitrary hexadecimal string of 8, 16, or 20 bytes. • Use the **authentication** keyword to specify that the key string is used with the ESP authentication transform. The **authentication** keyword is required only when the transform set includes an ESP authentication transform.
Step 15	**exit** **Example:** `RP/0/0/CPU0:router(config-new)# exit`	Exits profile configuration mode.

Step 16	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config)# end`
	or
	`RP/0/0/CPU0:router(config)# commit`
		Saves configuration changes.
		
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

	Command or Action	Purpose
Step 17	**show crypto ipsec sa** [**sa-id** | **peer** *ip-address* | **profile** *profile-name* | **detail** | **fvrf** *fvrf-name* | **ivrf** *ivrf-name* | **location** *location*] **Example:** `RP/0/0/CPU0:router# show crypto ipsec sa peer 172.19.72.120`	(Optional) Displays SA information based on the rack/slot/instance location. • Use the optional **detail** keyword to display additional dynamic SA information. The **detail** keyword is used only for software-based SAs. SAs that are configured under the tunnel-ipsec interface or crypto transport.
Step 18	**show crypto ipsec summary** **Example:** `RP/0/0/CPU0:router# show crypto ipsec summary`	(Optional) Displays IPSec summary information.

Configuring the DF Bit for the Encapsulating Header in IPSec Tunnels

This task configures the DF bit for the encapsulating header in IPSec tunnels. The DF bit configuration is also specified for both service-ipsec and service-gre interfaces.

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec df-bit** {**clear** | **set** | **copy**}
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ipsec df-bit** {**clear** | **set** | **copy**} **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec df-bit clear` or	Sets the DF bit for the encapsulating header in IPSec tunnels to all interfaces. You must specify at least one option for the **crypto ipsec df-bit** command. If no global setting is set, the default value is set to clear.

	Command or Action	Purpose

	Example:
	`RP/0/0/CPU0:router(config)# interface service-ipsec 5`
	`RP/0/0/CPU0:router(config-if)# crypto ipsec df-bit clear`
	`RP/0/0/CPU0:router(config-if)# crypto ipsec df-bit clear`
		Use the **crypto ipsec df-bit** command in global configuration mode and service-ipsec interface configuration mode.
		• (Optional) Use the **clear** keyword to specify that the outer IP header has the DF bit cleared and the router can fragment the packet to add the IPSec encapsulation.
		• (Optional) Use the **set** keyword to specify that the outer IP header has the DF bit set; however, the router can fragment the packet if the original packet had the DF bit cleared.
		• (Optional) Use the **copy** keyword to specify that the router looks in the original packet for the outer DF bit setting. The **copy** keyword is the default setting.

Step 3	**`end`**
	or
	`commit`
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the IPSec Antireplay Window: Expanding and Disabling

This section contains the following tasks:

- Configuring the IPSec Antireplay Window: Expanding and Disabling Globally, page SC-116 (optional)
- Disabling IPSec Replay Checking on a Crypto Profile, page SC-117 (optional)

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

Configuring the IPSec Antireplay Window: Expanding and Disabling Globally

This task configures the IPSec Antireplay Window: Expanding and Disabling globally.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec security-association replay window-size** [*N*]
3. c**rypto ipsec security-association replay disable**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ipsec security-association replay window-size** [*N*] **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec security-association replay window-size 256`	Sets the size of the SA replay window globally. • Use the *N* argument to specify the size of the window. Values are 64, 128, 256, 512, and 1024. This value becomes the default value. **Note** Configure this command or the **crypto ipsec security-association replay disable** command. The two commands are not used at the same time.

	Command or Action	Purpose
Step 3	**crypto ipsec security-association replay disable** **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec security-association replay disable`	Disables checking globally. **Note** Configure this command or the **crypto ipsec security-association replay window-size** command. The two commands are not used at the same time.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config)# end`
	or
	`RP/0/0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling IPSec Replay Checking on a Crypto Profile

This task disables replay checking on a crypto profile.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec profile** *name*
3. **set security-association replay disable**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ipsec profile** *name* **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec profile myprofile`	Creates or modifies a crypto profile entry and enters profile configuration mode. • Use the *name* argument to specify the name of an IPSec profile. The maximum length is 32 characters.
Step 3	**set security-association replay disable** **Example:** `RP/0/0/CPU0:router(config-myprofile)# set security-association replay disable`	Disables replay checking for a particular crypto profile.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-myprofile)# end`
	or
	`RP/0/0/CPU0:router(config-myprofile)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IPSec NAT Transparency

Network Address Translator (NAT) is automatically detected by the Cisco IPSec VPN SPA. If both VPN devices are NAT-T capable, NAT Transparency is automatically detected and automatically negotiated. No configuration steps are needed to enable IPSec NAT transparency.

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

Disabling IPSec NAT Transparency

This task disables NAT transparency if you already know that your network uses IPSec-awareness NAT (spi-matching scheme).

SUMMARY STEPS

1. **configure**
2. **crypto nat-transparency disable**
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto nat-transparency disable** **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec nat-transparency disable`	Disables the NAT transparency capability.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config)# end`
	or
	`RP/0/0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IPSec Security Association Idle Timers

To configure the IPSec Security Association Idle Timers feature, you must understand the following concepts and tasks:

- Lifetimes for IPSec Security Associations, page SC-121
- IPSec Security Association Idle Timers, page SC-121
- Configuring the IPSec SA Idle Timer Globally, page SC-121
- Configuring the IPSec SA Idle Timer for Each Crypto Profile, page SC-122

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

Lifetimes for IPSec Security Associations

Cisco IOS XR software currently allows the configuration of lifetimes for IPSec SAs. Lifetimes can be configured globally or for each crypto profile. Two lifetimes exist: a "timed" lifetime and a "traffic-volume" lifetime. A security association expires after the first of these lifetimes is reached.

IPSec Security Association Idle Timers

The IPSec SA idle timers are different from the global lifetimes for IPSec SAs. The expiration of the global lifetime is independent of peer activity. The IPSec SA idle timer allows SAs associated with inactive peers to be deleted before the global lifetime has expired.

If the IPSec SA idle timers are not configured, only the global lifetimes for IPSec SAs are applied. SAs are maintained until the global timers expire, regardless of peer activity.

Note If the last IPSec SA to a given peer is deleted because of idle timer expiration, the Internet Key Exchange (IKE) SA to that peer is also deleted.

Configuring the IPSec SA Idle Timer Globally

This task configures IPSec security association (SA) idle timers globally.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec security-association idle-time** *seconds*
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**crypto ipsec security-association idle-time** *seconds* **Example:** `RP/0/0/CPU0:router(config)# crypto ipsec security-association idle-time 600`	Configures the IPSec SA idle timer globally. • Use the *seconds* argument to specify the time, in seconds, that the idle timer allows an inactive peer to maintain an SA. Valid values for the *seconds* argument range from 600 to 86400.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config)# end`
	or
	`RP/0/0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the IPSec SA Idle Timer for Each Crypto Profile

This task configures the IPSec SA idle timer for a specified crypto profile. The idle timer configuration is applied to all SAs under the specified crypto profile. If no idle time is specified under the profile and idle time is configured in global mode, the idle time of the global mode is applied.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec profile** *name*
3. **set security-association idle-time** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto ipsec profile** *name*
	
	Example:
	`RP/0/0/CPU0:router(config)# crypto ipsec profile myprofile`
	`RP/0/0/CPU0:router(config-myprofile)#`
		Creates or modifies a crypto profile entry and enters profile configuration mode.
		• Use the *name* argument to specify the name of an IPSec profile. The maximum length is 32 characters.

Step 3	**set security-association idle-time** *seconds* **Example:** `RP/0/0/CPU0:router(config-myprofile)# set security-association idle-time 800`	Specifies the maximum time in which the current peer can be idle before the default peer is used. • Use the *seconds* argument to specify the number of seconds in which the current peer can be idle before the default peer is used. Valid values are 60 to 86400.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-myprofile)# end`
	or
	`RP/0/0/CPU0:router(config-myprofile)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling Prefragmentation for Cisco IPSec VPN SPAs

This section provides the following procedures to disable prefragmentation for Cisco IPSec VPN SPAs:

- Disabling Prefragmentation for service-ipsec Interfaces, page SC-124
- Disabling Prefragmentation for service-gre Interfaces, page SC-125

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

Disabling Prefragmentation for service-ipsec Interfaces

This task disables prefragmentation for service-ipsec interfaces.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec pre-fragmentation disable**
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose

Step 2	**crypto ipsec pre-fragmentation disable**
	
	Example:
	`RP/0/0/CPU0:router(config)# crypto ipsec pre-fragmentation disable`
	
	or
	
	Example:
	`RP/0/0/CPU0:router(config)# interface service-ipsec 5`
	`RP/0/0/CPU0:router(config-if)# crypto ipsec pre-fragmentation disable`
		Specifies the handling of fragmentation for the near-MTU-sized packets.
		• Use the **disable** keyword to disable the fragmentation of large packets before IPSec encapsulation.
		You can use the **crypto ipsec pre-fragmentation** command in global configuration mode or service-ipsec interface configuration mode.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config)# end`
	or
	`RP/0/0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling Prefragmentation for service-gre Interfaces

The fragmentation for a service-gre interface is done based on the original IP packet size, which does not include the GRE overhead.

This task disables prefragmentation for service-gre interfaces.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec pre-fragmentation disable**

3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto ipsec pre-fragmentation disable**
	
	Example:
	`RP/0/0/CPU0:router(config)# crypto ipsec pre-fragmentation disable`
	
	or
	
	Example:
	`RP/0/0/CPU0:router(config)# interface service-gre 500`
	`RP/0/0/CPU0:router(config-if)# crypto ipsec pre-fragmentation disable`
		Specifies the handling of fragmentation for the near-MTU-sized packets.
		• Use the **disable** keyword to disable the fragmentation of large packets before IPSec encapsulation.
		You can use the **crypto ipsec pre-fragmentation** command in global configuration mode or service-gre interface configuration mode.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config)# end`
	or
	`RP/0/0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Reverse-Route Injection in a Crypto Profile

This tasks shows how to configure reverse-route injection in a crypto profile.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec profile** *name*
3. **reverse-route**
4. **end**
   or
   commit
5. **show route** [**vrf** *vrf name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**crypto ipsec profile** *name*
	
	Example:
	`RP/0/0/CPU0:router(config)# crypto ipsec profile myprofile`
	`RP/0/0/CPU0:router(config-myprofile)#`
		Creates or modifies a crypto profile entry and enters profile configuration mode.
		• Use the *name* argument to specify the name of an IPSec profile. Maximum length is 32 characters.

Step 3	**reverse-route** **Example:** `RP/0/0/CPU0:router(config-myprofile)# reverse-route`	Creates source proxy information for a crypto profile entry.

	Command or Action	Purpose

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-myprofile)# end`
	or
	`RP/0/0/CPU0:router(config-myprofile)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show route** [**vrf** *vrf name*] **Example:** `RP/0/0/CPU0:router#`	Displays the proxy information that has been added as static routes. The SAs, which are created on these interfaces, are also called software-based SAs.

Configuring IPSec Failure History Table Size

This task changes the size of the failure history table.

Note This IPSec feature is supported only on the Cisco IPSec VPN SPA.

SUMMARY STEPS

1. **configure**
2. **crypto mib ipsec flowmib history failure size** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto mib ipsec flowmib history failure size** *number* **Example:** `RP/0/0/CPU0:router(config)# crypto mib ipsec flowmib history failure size 140`	Sets the size of the failure history table.

How to Implement IPSec Network Security for Locally Sourced and Destined Traffic

Locally sourced and terminated traffic are evaluated against IPSec profiles that are attached to tunnel-ipsec interfaces or crypto transport.

Note

- Multiple profiles can be attached to a tunnel-ipsec interface or crypto transport.
- For locally sourced traffic or terminated traffic, we discourage the use of the **any** keyword to specify source or destination addresses in the crypto profiles, which are attached to the tunnel-ipsec interface or transport. This recommendation is only for locally sourced traffic for VPN transit traffic. You can encrypt all the traffic going through the interface. Therefore, ACLs in profiles, which are attached to service-ipsec interfaces, can use the **any** keyword).

This section contains the following procedures:

- The any Keyword in Crypto Access Lists, page SC-129
- Applying Crypto Profiles to tunnel-ipsec Interfaces, page SC-130
- Applying Crypto Profiles to Crypto Transport, page SC-131

The any Keyword in Crypto Access Lists

When you create crypto access lists, using the **any** keyword could cause problems. We discourage the use of the **any** keyword to specify source or destination addresses. The **any** keyword is relevant only to locally sourced or terminated traffic.

No concept of default access lists exists for IPSec.

The **permit any any** statement is strongly discouraged, because it causes all outbound traffic to be protected (and all protected traffic to be sent to the peer specified in the corresponding crypto profile entry) and requires protection for all inbound traffic. Then, all inbound packets that lack IPSec protection are silently dropped, including packets for routing protocols, NTP, echo, echo response, and so on.

Be sure to define which packets to protect. If you *must* use the **any** keyword in a **permit** statement, you must preface that statement with a series of **deny** statements to filter any traffic (that would otherwise fall within that **permit** statement) that you do not want to be protected.

Applying Crypto Profiles to tunnel-ipsec Interfaces

This task applies a crypto IPsec profile to a tunnel-ipsec interface.

You must apply a crypto profile to each tunnel-ipsec interface through which IPSec traffic flows. Applying the crypto profile set to a tunnel-ipsec interface instructs the router to evaluate all the interface's traffic against the crypto profile set and to use the specified policy during connection or SA negotiation on behalf of traffic to be protected by crypto.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ipsec** *interface-number*
3. **profile** *profile-name*
4. **tunnel source** *ip-address*
5. **tunnel destination** *ip-address*
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure interface`	Enters global configuration mode.
Step 2	**interface tunnel-ipsec** *interface-number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-ipsec 0`	Identifies the IPSec interface to which the crypto profile is attached. You can use the **interface tunnel-ipsec** command to enter tunnel-ipsec interface configuration mode.
Step 3	**profile** *profile-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# profile sample1`	Specifies the crypto profile to use in IPSec processing. • The same crypto profile cannot be shared in different IPSec modes.
Step 4	**tunnel source** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-if)# tunnel source 10.0.0.2`	Specifies the tunnel source IP address. • This command is required for both static and dynamic profiles.

	Command or Action	Purpose
Step 5	**tunnel destination** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-if)# tunnel destination 10.0.0.5`	Specifies the tunnel destination IP address. • This command is not required if the profile is dynamic.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Applying Crypto Profiles to Crypto Transport

This task applies a crypto profile to crypto transport.

You need to apply a crypto profile to transport mode to make the profile active. Applying the crypto profile set to transport instructs the router to evaluate all of the locally sourced traffic against the crypto profile set and use the specified policy during connection or SA negotiation on behalf of traffic to be protected by crypto.

SUMMARY STEPS

1. **configure**
2. **crypto ipsec transport**
3. **profile** *profile-name*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**crypto ipsec transport** **Example:** `RP/0/RP0/CPU0:router(config)# crypto ipsec transport`	Enters IPSec transport configuration mode. • In the IPSec transport configuration mode, IPSec protects the Upper Layer Protocol (ULP) header and the payload. IPSec transport configuration mode is used when security is desired end to end. That is, security endpoints are the same as host endpoints.
Step 3	**profile** *profile-name* **Example:** `RP/0/RP0/CPU0:router(config-transport)# profile sample2`	Specifies the crypto profile to use in IPSec processing.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-transport)# end`
	or
	`RP/0/RP0/CPU0:router(config-transport)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

How to Implement IPSec Network Security for VPNs

This section contains the following procedures:

- Configuring IPSec Virtual Interfaces, page SC-133
- Configuring the Default Path Maximum Transmission Unit for the SA, page SC-139

Configuring IPSec Virtual Interfaces

These tasks configure IPSec virtual interfaces:

- Configuring Static IPSec Virtual Interfaces, page SC-133
- Configuring IPSec-Protected GRE Virtual Interfaces, page SC-136

Configuring Static IPSec Virtual Interfaces

This task configures static IPSec service virtual interfaces (SVIs).

SUMMARY STEPS

1. **configure**
2. **interface service-ipsec** *number*
3. **profile** *profile-name*
4. **tunnel source** *ip-address*
5. **tunnel destination** *ip-address*
6. **tunnel vrf** *vrf-name*
7. **vrf** *vrf-name*
8. **ipv4 address** *ipv4-address mask* [**secondary**]
9. **service-location preferred-active** *location* [**preferred-standby** *location* [**auto-revert**]]
10. **end**
    or
    commit
11. **show route** [**vrf** *vrf name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**interface service-ipsec** *number*
	
	Example:
	`RP/0/0/CPU0:router(config)# interface service-ipsec 2`
	`RP/0/0/CPU0:router(config-if)#`
		Creates a static IPSec SVI.
		You can use the **interface service-ipsec** command to enter service-ipsec interface configuration mode.

Step 3	**profile** *profile-name* **Example:** `RP/0/0/CPU0:router(config-if)# profile ipsec_profa`	Specifies the crypto profile to use for IPSec processing. • Use the *profile-name* argument to define the previous crypto profile to use. The character range is from 1 to 32 characters.

	Command or Action	Purpose
Step 4	**tunnel source** {*ip-address*} **Example:** `RP/0/0/CPU0:router(config-if)# tunnel source 172.19.72.92`	Specifies the source address for a tunnel-ipsec interface. • Use the *ip-address* argument to set the IP address to use as the source address for packets in the tunnel.
Step 5	**tunnel destination** *ip-address* **Example:** `RP/0/0/CPU0:router(config-if)# tunnel destination 172.19.72.120`	Identifies the IP address of the tunnel destination. • Use the *ip-address* argument to set the IP address of the host destination. If the IPSec profile is a dynamic, the tunnel destination should not be configured.
Step 6	**tunnel vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-if)# tunnel vrf internet`	Associates a VRF instance with the tunnel source or destination of the interfaces. The tunnel VRF specifies in which VRF the tunneled traffic is forwarded (FVRF). Tunnel VRF is not required if FVRF is the global VRF. • Use the *vrf-name* argument to assign the name of a VRF.
Step 7	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-if)# vrf vpn_a`	Assigns a VRF to the interface. VRF is specified to clear traffic that is forwarded for the internal VRF (IVRF). In addition, VRF is not required if IVRF is a global VRF. • Use the *vrf-name* argument to assign the name of a VRF.
Step 8	**ipv4 address** *ipv4-address mask* [**secondary**] **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0`	Sets a primary or secondary IPv4 address for an interface, for example, POS interface. • Use the *ipv4-address* argument to set the IPv4 address. • Use the *mask* argument to set the mask for the associated IP subnet. The network mask is specified in either of two ways: – The network mask is a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means the corresponding address bit belongs to the network address. – The network mask is indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address. • (Optional) Use the **secondary** keyword to specify that the configured address is a secondary IPv4 address. If this keyword is omitted, the configured address is the primary IPv4 address.

	Command or Action	Purpose
Step 9	**service-location preferred-active** *location* [**preferred-standby** *location* [**auto-revert**]] **Example:** `RP/0/0/CPU0:router(config-if)# service-location preferred-active 0/0/0 preferred-standby 0/1/0`	Specifies both active and standby locations for the interface. • Use the **preferred-active** keyword to specify that the SPA in this location serves all traffic going through the interface. The location argument is expressed in *rack/slot/module* notation. • (Optional) Use the **preferred-standby** keyword to specify that if a SPA fails, the interface is served by the SPA in this location. The location argument is expressed in *rack/slot/module* notation. • (Optional) Use the **auto-revert** keyword to revert to the preferred-active location if the **auto-revert** keyword is configured. **Note** The **auto-revert** keyword is specified only if the **preferred-standby** keyword with the *location* argument is configured.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show route** [**vrf** *vrf name*] **Example:** `RP/0/0/CPU0:router# show route vrf`	Displays the proxy information that was added as static routes.

Configuring IPSec-Protected GRE Virtual Interfaces

This task configures IPSec-protected GRE service virtual interfaces.

SUMMARY STEPS

1. **configure**
2. **interface service-gre** *number*
3. **profile** *profile-name*
4. **tunnel source** {*ip-address*}
5. **tunnel destination** *ip-address*
6. **tunnel vrf** *vrf-name*
7. **vrf** *vrf-name*
8. **ipv4 address** *ipv4-address mask* [**secondary**]
9. **service-location preferred-active** *location* [**preferred-standby** *location*] [**auto-revert**]
10. **end**
    or
    commit
11. **show route** [**vrf** *vrf name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**interface service-gre** *number*
	
	Example:
	`RP/0/0/CPU0:router(config)# interface service-gre 2`
	`RP/0/0/CPU0:router(config-if)#`
		Creates a GRE service virtual interface.
		You can use the **interface service-gre** command to enter service-gre interface configuration mode

Step 3	**profile** *profile-name* **Example:** `RP/0/0/CPU0:router(config-if)# profile ipsec_profa`	Specifies the crypto profile to use for IPSec processing. For the service-gre interface, the IPSec profile must be static. • Use the *profile-name* argument to define the previous crypto profile to use. The character range is from 1 to 32 characters.
Step 4	**tunnel source** {ip-address} **Example:** `RP/0/0/CPU0:router(config-if)# tunnel source 172.19.72.92`	Specifies the source address for a tunnel-ipsec interface. • Use the *ip-address* argument to set the IP address to use as the source address for packets in the tunnel.

	Command or Action	Purpose
Step 5	**tunnel destination** *ip-address* **Example:** `RP/0/0/CPU0:router(config-if)# tunnel destination 172.19.72.120`	Identifies the IP address of the tunnel destination. • Use the *ip-address* argument to set the IP address of the host destination. If dynamic, the destination IP address is optional.
Step 6	**tunnel vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-if)# tunnel vrf internet`	Associates a VRF instance with the tunnel source or destination of the interfaces. The tunnel VRF specifies in which VRF the tunneled traffic is forwarded (FVRF). Tunnel VRF is not required if FVRF is the global VRF. • Use the *vrf-name* argument to assign the name of a VRF.
Step 7	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-if)# vrf vpn_a`	Assigns a VRF to the interface. VRF is specified to clear traffic that is forwarded for the internal VRF (IVRF). In addition, VRF is not required if IVRF is a global VRF. • Use the *vrf-name* argument to assign the name of a VRF.
Step 8	**ipv4 address** *ipv4-address mask* [**secondary**] **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0`	Sets a primary or secondary IPv4 address for an interface, for example, a POS interface. • Use the *ipv4-address* argument to set the IPv4 address. • Use the *mask* argument to set the mask for the associated IP subnet. The network mask is specified in either of two ways: – The network mask is a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address. – The network mask is indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are the network address. • (Optional) Use the **secondary** keyword to specify that the configured address is a secondary IPv4 address. If this keyword is omitted, the configured address is the primary IPv4 address.

	Command or Action	Purpose
Step 9	**service-location preferred-active** *location* [**preferred-standby** *location* [**auto-revert**] **Example:** `RP/0/0/CPU0:router(config-if)# service-location preferred-active 0/0/0 preferred-standby 0/1/0`	Specifies both active and standby locations for the interface. • Use the **preferred-active** keyword to specify that the SPA in this location serves all traffic going through the interface. The location argument is expressed in *rack/slot/module* notation. • (Optional) Use the **preferred-standby** keyword to specify that if a SPA fails, the interface is served by the SPA in this location. The location argument is expressed in *rack/slot/module* notation. • (Optional) Use the **auto-revert** keyword to revert to the preferred-active location if the **auto-revert** keyword is configured. **Note** The **auto-revert** keyword is specified only if the **preferred-standby** keyword with the *location* argument is configured.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show route** [**vrf** *vrf name*] **Example:** `RP/0/0/CPU0:router# show route vrf`	Displays the proxy information that was added as static routes.

Configuring the Default Path Maximum Transmission Unit for the SA

This task configures the default path maximum transmission unit (MTU) for the SA.

SUMMARY STEPS

1. **configure**
2. **interface service-ipsec** *number*
3. **crypto ipsec pmtu** *pmtu*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**interface service-ipsec** *number*
	
	Example:
	`RP/0/0/CPU0:router(config)# interface service-ipsec 2`
	`RP/0/0/CPU0:router(config-if)#`
		Creates a static IPSec SVI.
		You can use the **interface service-ipsec** command to enter service-ipsec interface configuration mode.

	Command or Action	Purpose
Step 3	**crypto ipsec pmtu** *pmtu* **Example:** `RP/0/0/CPU0:router(config-if)# crypto ipsec pmtu 1500`	Specifies the default path MTU for the SAs that are created under the interface. • Use the *pmtu* argument to specify the value of MTU in bytes. The range is from 68 to 9216.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing IPSec Network Security for Locally Sourced Traffic and Destined Traffic

This section provides the following configuration examples:

- Configuring a Static Profile and Attaching to a Tunnel-ipsec Interface: Example, page SC-140
- Configuring a Dynamic Profile and Attaching to a Tunnel-ipsec Interface: Example, page SC-141
- Configuring a Static Profile and Attaching to Transport: Example, page SC-142

Configuring a Static Profile and Attaching to a Tunnel-ipsec Interface: Example

The following example shows a minimal IPSec configuration where a static crypto profile is created and attached to a tunnel-ipsec interface.

An IPSec access list named sample1 defines which traffic to protect:

```
ipv4 access-list sample1 permit ip 10.0.0.0 0.0.0.255 10.2.2.0 0.0.0.255
```

A transform set defines how the traffic is protected. In this example, transform set myset1 uses Data Encryption Standard (DES) encryption and Secure Hash Algorithm (SHA) for data packet authentication:

```
crypto ipsec transform-set myset1
   transform esp-des esp-sha
```

Another transform set example is myset2, which uses 3DES encryption and the Message Digest 5 (MD5) (Hashed Message Authentication Code [HMAC] variant) algorithm for data packet authentication:

```
crypto ipsec transform-set myset2
   transform esp-3des esp-md5-hmac
```

A crypto profile named toRemoteSite is created and joins the IPSec access list and transform set:

```
crypto ipsec profile toRemoteSite
    match sample1 transform-set myset1
    end
```

The toRemoteSite crypto profile is then applied to a tunnel-ipsec interface:

```
interface tunnel-ipsec0
    profile toRemoteSite
    tunnel source 10.0.0.2
    tunnel destination 10.0.0.5
```

Configuring a Dynamic Profile and Attaching to a Tunnel-ipsec Interface: Example

The following example shows a minimal IPSec configuration where a dynamic crypto profile is created and attached to a tunnel-ipsec interface.

An IPSec access list named sample2 defines which traffic to protect:

```
ipv4 access-list sample2 permit ip 10.0.0.0 0.0.0.255 10.2.2.0 0.0.0.255
```

A transform set defines how the traffic is protected. In this example, transform set myset2 uses DES encryption and SHA for data packet authentication:

```
crypto ipsec transform-set myset2
   transform esp-des esp-sha
```

Another transform set example is myset3, which uses 3DES encryption and MD5 (HMAC variant) for data packet authentication:

```
crypto ipsec transform-set myset3
   transform esp-3des esp-md5-hmac
```

A dynamic crypto profile named toRemoteSite is created and joins the IPSec access list and transform set:

```
crypto ipsec profile toRemoteSite
    match sample2 transform-set myset3
    set type dynamic discover
    end
```

The toRemoteSite profile is applied to a tunnel-ipsec interface:

```
interface tunnel-ipsec0
    profile toRemoteSite
    tunnel source 10.0.0.2
```

The tunnel destination is not required when the profile is dynamic.

Configuring a Static Profile and Attaching to Transport: Example

The following example shows a minimal IPSec configuration in which a static profile is created and attached to a transport.

An IPSec access list named sample3 defines which traffic to protect:

```
ipv4 access-list sample3 permit ip 10.0.0.0 0.0.0.255 10.2.2.0 0.0.0.255
```

A transform set defines how the traffic is protected. In this example, transform set myset1 uses DES encryption and SHA for data packet authentication:

```
crypto ipsec transform-set myset1
   transform esp-des esp-sha
```

Another transform set example is myset2, which uses 3DES encryption and the MD5 (HMAC variant) for data packet authentication:

```
crypto ipsec transform-set myset2
   transform esp-3des esp-md5-hmac
```

A crypto profile named toRemoteSite is created and joins the IPSec access list and transform set:

```
crypto ipsec profile toRemoteSite
    match sample3 transform-set myset2
    end
```

The toRemoteSite profile is applied to a transport:

```
crypto ipsec transport
    profile toRemoteSite
    end
```

Configuration Examples for an IPSec Network with a Cisco IPSec VPN SPA

This section provides the following configuration examples:

- Configuring IPSec for a VRF-aware Service-ipsec Interface: Example, page SC-142
- Configuring a Service-gre Interface: Example, page SC-145

Configuring IPSec for a VRF-aware Service-ipsec Interface: Example

The following example shows an IPSec configuration of a VRF-aware service-ipsec interface with a crypto IPSec profile that uses RRI.

The **interface service-ipsec** command is set to 1 and is part of the customer_1 VRF. FVRF is the global VRF (default). Clear traffic is coming from customer_1 VRF with a source IP address 100.0.1.0/24 and is destined to 30.0.1.0/24, which is encrypted and sent over to the global VRF. Respectively, the encrypted traffic from 30.0.1.0/24 is destined to 100.0.1.0/24 and is encrypted on the remote site or host and decrypted on the router.

Configuring VRF

```
vrf customer_1
 address-family ipv4 unicast
```

```
  import route-target
   100:1000
  !
  export route-target
   100:1000
  !
 !
!
```

Configuring ACL That Is Used by the IPSec Profile

```
ipv4 access-list acl1
 10 permit ipv4 100.0.1.0 0.0.0.255 30.0.1.0 0.0.0.255
!
```

Configuring the Service-ipsec Interface

```
interface service-ipsec1
 vrf customer_1  <------------- IVRF
 ipv4 address 40.40.41.41 255.255.255.0
 profile prof1   <--------- the ipsec profile
 tunnel source 4.0.1.1
 tunnel destination 5.0.1.1
 service-location preferred-active 0/1/1 preferred-standby 0/2/0 <----------- The IPSec
SPA is located on the 0/1/1 and the standby SPA on  0/2/0
!
```

Configuring IKE

```
crypto isakmp
crypto isakmp policy 1
 authentication pre-share
 encryption 3des
 lifetime 86400
!
crypto keyring kr1 vrf default
 pre-shared-key address 5.0.1.1 255.255.255.255 key aBrAkAdAbRa

crypto isakmp profile a_prof
 keyring kr1
 match identity address 5.0.1.1/32 vrf default
  set interface service-ipsec1
 !
```

Configuring IPSec

The following example shows that the transform-set is set to esp-256-aes:

```
crypto ipsec transform-set ts1
 transform esp-256-aes
!
```

The following example shows that the IPSec profile uses acl1 as the traffic proxy and transform-set is ts1. In addition, RRI is configured.

```
crypto ipsec profile prof1
 set pfs group1
 set type static
 match acl1 transform-set ts1
 reverse-route
```

```
!
```

The following example shows that the IPSec SA is created from the **show crypto ipsec summary** command and **show crypto ipsec sa** command:

```
RP/0/RP0/CPU0:router# show crypto ipsec summary

# * Attached to a transform indicates a bundle

# Active IPSec Sessions: 1

SA    Local Peer      Remote Peer     FVRF      Profile   Transform    Lifetime
-------------------------------------------------------------------------------
502   4.0.1.1         5.0.1.1         default   prof1     esp-256-aes  3600/4194303

RP/0/RP0/CPU0:router# show crypto ipsec sa

SA id:          502
Node id:        0/1/1 0/2/0
SA Type:        ISAKMP
interface:      service-ipsec1
profile  :      prof1
local  ident (addr/mask/prot/port) : (100.0.1.0/255.255.255.0/0/0)
remote ident (addr/mask/prot/port) : (30.0.1.0/255.255.255.0/0/0)
local crypto endpt: 4.0.1.1, remote crypto endpt: 5.0.1.1, vrf default

 #pkts tx         :0                      #pkts rx         :0
 #bytes tx        :0                      #bytes rx        :0
 #pkts encrypt    :0                      #pkts decrypt    :0
 #pkts digest     :0                      #pkts verify     :0
 #pkts encrpt fail:0                      #pkts decrpt fail:0
 #pkts digest fail:0                      #pkts verify fail:0
 #pkts replay fail:0
 #pkts tx errors  :0                      #pkts rx errors  :0

outbound esp sas:
        spi: 0x3482d5c8(880989640)
        transform: esp-256-aes
        in use settings = Tunnel
        sa agreed lifetime: 3600s, 4194303kb
        sa timing: remaining key lifetime (sec/kb): (3525/4194303)
        sa DPD disabled
        sa idle timeout: disable, 0s
        sa anti-replay (HW accel): enable, window 64
inbound esp sas:
        spi: 0x3c9869ee(1016621550)
        transform: esp-256-aes
        in use settings = Tunnel
        sa agreed lifetime: 3600s, 4194303kb
        sa timing: remaining key lifetime (sec/kb): (3525/4194303)
        sa DPD disabled
        sa idle timeout: disable, 0s
        sa anti-replay (HW accel): enable, window 64
```

The following example shows that RRI was configured so the proxy is added to the routing table of the VRF:

```
RP/0/RP0/CPU0:router# show route vrf customer_1

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
```

```
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

S    30.0.1.0/24 is directly connected, 00:02:09, service-ipsec1
C    40.40.41.0/24 is directly connected, 00:02:09, service-ipsec1
L    40.40.41.41/32 is directly connected, 00:02:09, service-ipsec1
C    100.100.100.0/24 is directly connected, 00:01:26, GigabitEthernet0/0/0/3
L    100.100.100.1/32 is directly connected, 00:01:26, GigabitEthernet0/0/0/3
```

The following example shows that the **interface service-ipsec** command is set to 1 and is part of the customer_1 VRF:

```
RP/0/RP0/CPU0:router# show crypto ipsec interface service-ipsec 1

--------------- IPSec interface ----------------
Interface service-ipsec1, mode Tunnel, intf_handle 0x5000180
Locations 0/1/1 0/2/0,  VRF customer_1 (60000002)
Number of profiles 1, number of flows 1
Tunnel: source 4.0.1.1, destination 5.0.1.1, tunnel VRF default
DF-bit: copy, pre-fragmentation enable
default pmtu: 9216
1 connected flows:
502
```

Configuring a Service-gre Interface: Example

The following example shows a basic configuration of a service-gre interface and an IPSec SA that is created on the interface.

Configuring the Transform-set to Use Transport Mode

```
crypto ipsec transform-set tsfm2
 transform esp-3des esp-md5-hmac
 mode transport
!
```

Configuring the IPSec Profile to Use the Set Transform-set Format

```
crypto ipsec profile gre
 set transform-set tsfm2
!
```

Configuring the Service-gre Interface

```
interface service-gre1
 ipv4 address 11.2.6.6 255.255.255.0
 profile gre
 tunnel source 50.50.50.2
 tunnel destination 40.40.40.2
 service-location preferred-active 0/1/1
!
```

The following example shows the sample output from the **show crypto ipsec summary** command:

```
RP/0/RP0/CPU0:router# show crypto ipsec summary

# * Attached to a transform indicates a bundle
```

```
# Active IPSec Sessions: 2

SA    Local Peer      Remote Peer     FVRF      Profile   Transform    Lifetime
-------------------------------------------------------------------------------
503   50.50.50.2      40.40.40.2      default   gre       esp-3des esp 120/4194303
```

The following example shows that the service-gre interface is set to 1 with a profile gre:

```
RP/0/RP0/CPU0:router# show crypto ipsec sa 503

SA id:          503
Node id:        0/1/1
SA Type:        ISAKMP
interface:      service-gre1
profile  :      gre
local  ident (addr/mask/prot/port) : (50.50.50.2/255.255.255.255/47/0)
remote ident (addr/mask/prot/port) : (40.40.40.2/255.255.255.255/47/0)
local crypto endpt: 50.50.50.2, remote crypto endpt: 40.40.40.2, vrf default

 #pkts tx         :0                      #pkts rx         :0
 #bytes tx        :0                      #bytes rx        :0
 #pkts encrypt    :0                      #pkts decrypt    :0
 #pkts digest     :0                      #pkts verify     :0
 #pkts encrpt fail:0                      #pkts decrpt fail:0
 #pkts digest fail:0                      #pkts verify fail:0
 #pkts replay fail:0
 #pkts tx errors  :0                      #pkts rx errors  :0

outbound esp sas:
        spi: 0x5aeffcbd(1525677245)
        transform: esp-3des esp-md5-hmac
        in use settings = Transport
        sa agreed lifetime: 120s, 4194303kb
        sa timing: remaining key lifetime (sec/kb): (108/4194303)
        sa DPD disabled
        sa idle timeout: disable, 0s
        sa anti-replay (HW accel): enable, window 64
inbound esp sas:
        spi: 0x54373dd3(1412906451)
        transform: esp-3des esp-md5-hmac
        in use settings = Transport
        sa agreed lifetime: 120s, 4194303kb
        sa timing: remaining key lifetime (sec/kb): (108/4194303)
        sa DPD disabled
        sa idle timeout: disable, 0s
        sa anti-replay (HW accel): enable, window 64
```

The following example shows that the **interface service-gre** command is set to 1:

```
RP/0/RP0/CPU0:router# show crypto ipsec interface service-gre 1

--------------- IPSec interface ----------------
Interface service-gre1, mode Transport, intf_handle 0x5000880
Locations 0/1/1,  VRF default (60000000)
Number of profiles 1, number of flows 1
Tunnel: source 50.50.50.2, destination 40.40.40.2, tunnel VRF default
DF-bit: copy, pre-fragmentation enable
default pmtu: 9216
1 connected flows:
503
```

Additional References

The following sections provide references related to implementing IPSec network security.

Related Documents

Related Topic	Document Title
IPSec network security commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*IPSec Network Security Commands on Cisco IOS XR Software* module in *Cisco IOS XR System Security Command Reference*, Release 3.5
Internet Key Exchange (IKE) security protocol commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Internet Key Exchange Security Protocol Commands on Cisco IOS XR Software* module in *Cisco IOS XR System Security Command Reference*, Release 3.5
IPSec in Quality of Service (QoS)	*Cisco IOS XR Modular Quality of Service Configuration Guide*, Release 3.5
MPLS distribution protocol	*Cisco IOS XR Multiprotocol Label Switching Configuration Guide*, Release 3.5

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
RFC 2401	*Security Architecture for the Internet Protocol*
RFC 2402	*IP Authentication Header*
RFC 2403	*The Use of HMAC-MD5-96 within ESP and AH*
RFC 2404	*The Use of HMAC-SHA-1-96 within ESP and AH*
RFC 2405	*The ESP DES-CBC Cipher Algorithm With Explicit IV*
RFC 2406	*IP Encapsulating Security Payload (ESP)*
RFC 2407	*The Internet IP Security Domain of Interpretation for ISAKMP*
RFC 2408	*Internet Security Association and Key Management Protocol (ISAKMP)*
RFC 2409	*The Internet Key Exchange (IKE)*

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

Implementing Secure Shell on Cisco IOS XR Software

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 SSH are available: SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2). SSHv1 uses Rivest, Shamir, and Adelman (RSA) keys and SSHv2 uses Digital Signature Algorithm (DSA) keys. Cisco IOS XR software supports both SSHv1 and SSHv2.

This module describes the tasks that you need to implement Secure Shell on your Cisco IOS XR network.

Note For a complete description of the Secure Shell commands used in this chapter, see the *Secure Shell Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference* publication. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing Secure Shell on Cisco Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.3.0	The **ssh server v2** command was added.
Release 3.4.0	No modification.
Release 3.5.0	No modification.

Contents

- Prerequisites to Implementing Secure Shell, page SC-150
- Restrictions for Implementing Secure Shell, page SC-150
- Information About Implementing Secure Shell, page SC-151
- How to Implement Secure Shell, page SC-152

- Configuration Examples for Implementing Secure Shell, page SC-156
- Additional References, page SC-156

Prerequisites to 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 for security commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- Download the required image on your router. The SSH server and SSH client require you to have a a crypto package (data encryption standard [DES], 3DES and AES) from Cisco downloaded on your router.
- 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 *Authentication, Authorization, and Accounting Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference* publication and *Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide* publication.
- AAA authentication and authorization must be configured correctly for Secure Shell File Transfer Protocol (SFTP) to work.

Restrictions for Implementing Secure Shell

The following are some basic SSH restrictions and limitations of the SFTP feature:

- In order for an outside client to connect to the router, the router needs to have an RSA (for SSHv1) or DSA (for SSHv2) key pair configured. 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: DSA and RSA keys are not required because SFTP only operates in client mode.
- For SFTP to work properly, the remote SSH server must enable 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 available in SSH clients is not supported in the SSH server for Cisco IOS XR software.
- Execution shell and SFTP are the only applications supported.
- The SFTP client does not support remote filenames containing wildcards (*, ?, []). The user must issue the **sftp** command multiple times or list all of the source files from the remote host to download them on to the router. For uploading, the router SFTP client can support multiple files specified using a wildcard provided that the issues mentioned in the first through third bullets in this section are resolved.

- 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.

Information About Implementing Secure Shell

To implement SSH, you should understand the following concepts:

- SSH Server, page SC-151
- SSH Client, page SC-151
- SFTP Feature Overview, page SC-151
- AAA Feature, page SC-152

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 DES, 3DES, message digest algorithm 5 (MD5), SHA1, and password authentication. User authentication is performed like that in the Telnet session to the router. The user authentication mechanisms supported for SSH are RADIUS, TACACS+, and the use of locally stored usernames and passwords.

SFTP Feature Overview

SSH includes support for SFTP, which is a feature that provides a secure and authenticated method for copying router configuration or router image files.

SFTP is the new, standard file transfer protocol introduced in SSHv2. 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 EXEC mode.

AAA Feature

AAA is a suite of network security services that provide the primary framework through which access control can be set up on your Cisco router or access server. For more information on AAA, see the *Authentication, Authorization, and Accounting Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference* publication and the *Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide* publication.

How to Implement Secure Shell

To configure SSH, perform the tasks described in the following sections:

- Configuring SSH, page SC-152 (required)
- Configuring the SSH Client, page SC-154 (required)

Configuring SSH

Perform this task to configure SSH.

Note For SSHv1 configuration, Step 1 to Step 4 are required. For SSHv2 configuration, Step 1 to Step 4 are optional.

SUMMARY STEPS

1. **configure**
2. **hostname** *hostname*
3. **domain name** *domain-name*
4. **exit**
5. **crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*]
6. **crypto key generate dsa**
7. **configure**
8. **ssh timeout** *seconds*
9. **ssh server**
   or
   ssh server v2 (optional)
10. **end**
    or
    commit
11. **show ssh**
12. **show ssh session details**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**hostname** *hostname* **Example:** `RP/0/RP0/CPU0:router(config)# hostname router1`	Configures a hostname for your router.
Step 3	**domain name** *domain-name* **Example:** `RP/0/RP0/CPU0:router(config)# domain name cisco.com`	Defines a default domain name that the software uses to complete unqualified host names.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config)# exit`	Exits global configuration mode, and returns the router to EXEC mode.
Step 5	**crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*] **Example:** `RP/0/RP0/CPU0:router# crypto key generate rsa general-keys`	Generates an RSA key pair. • To delete the RSA key pair, use the **crypto key zeroize rsa** command. This command is used for SSHv1 only.
Step 6	**crypto key generate dsa** **Example:** `RP/0/RP0/CPU0:router# crypto key generate dsa`	Enables the SSH server for local and remote authentication on the router. • The recommended minimum modulus size is 1024 bits. • Generates a DSA key pair. To delete the DSA key pair, use the **crypto key zeroize dsa** command. This command is used only for SSHv2.
Step 7	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 8	**ssh timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config)# ssh timeout 60`	(Optional) Configures the timeout value for user authentication to AAA. • If the user fails to authenticate itself to AAA within the configured time, the connection is aborted. • If no value is configured, the default value of 30 seconds is used. The range is from 5 to 120.

	Command or Action	Purpose

Step 9	**ssh server**
	or
	ssh server v2
	
	Example:
	`RP/0/RP0/CPU0:router(config)# ssh server`
	or
	`RP/0/RP0/CPU0:router(config)# ssh server v2`
		Brings up an SSH server.
		• To bring down an SSH server, use the **no ssh server** command.
		(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 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show ssh** **Example:** `RP/0/RP0/CPU0:router# show ssh`	(Optional) Displays all of the incoming and outgoing SSHv1 and SSHv2 connections to the router.
Step 12	**show ssh session details** **Example:** `RP/0/RP0/CPU0:router# show ssh session details`	(Optional) Displays a detailed report of the SSHv2 connections to and from the router.

Configuring the SSH Client

Perform this task to configure an SSH client.

SUMMARY STEPS

1. **configure**
2. **ssh client knownhost** *device:/filename*

3. **exit**
4. **ssh** {*ipv4-address* | *ipv6-address* | *hostname*} [**username** *user-id* | **cipher des** | **source-interface** *type instance*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ssh client knownhost** *device*:/*filename* **Example:** `RP/0/RP0/CPU0:router(config)# ssh client knownhost slot0:/server_pubkey`	(Optional) Enables the feature to authenticate and check the server public key (pubkey) at the client end. • The complete path of the filename is required. The colon (:) and slash mark (/) are also required.
Step 3	**exit** **Example:** `RP/0/RP0/CPU0:router(config)# exit`	Exits global configuration mode, and returns the router to EXEC mode.
Step 4	**ssh** {*ipv4-address* | *ipv6-address* | *hostname*} [**username** *user-id* | **cipher des** | **source-interface** *type instance*] **Example:** `RP/0/RP0/CPU0:router# ssh remotehost username user1234`	Enables an outbound SSH connection. • 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. • If the *hostname* argument is used and the host has both IPv4 and IPv6 addresses, the IPv6 address is used.

Troubleshooting Tips

- If you are using SSHv1 and your SSH connection is being rejected, you have not successfully generated an RSA key pair for your router. 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 enable the SSH server.
- If you are using SSHv2 and your SSH connection is being rejected, you have not successfully generated a DSA key pair for your router. Make sure that you have specified a hostname and domain. Then use the **crypto key generate dsa** command to generate a DSA key pair and enable the SSH server.
- When configuring the 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** global configuration command.
  - No domain specified

You must configure a host domain for the router using the **domain-name** global configuration command.

- The number of allowable SSH connections is limited to the maximum number of virtual terminal lines configured for the router. Each SSH connection uses a vty resource.
- SSH uses either local security or the security protocol that is configured through AAA on your router for user authentication. When configuring AAA, you must ensure that the console is not running under AAA by applying a keyword in the global configuration mode to disable AAA on the console.

Configuration Examples for Implementing Secure Shell

This section provides the following configuration example:

- Configuring Secure Shell: Example, page SC-156

Configuring Secure Shell: Example

The following example shows how to configure SSHv2 by creating a hostname, defining a domain name, enabling the SSH server for local and remote authentication on the router by generating a DSA key pair, bringing up the SSH server, and saving the configuration commands to the running configuration file.

After SSH has been configured, the SFTP feature is available on the router.

```
configure
    hostname router1
    domain name cisco.com
    exit
crypto key generate dsa
configure
    ssh server
    end
```

Additional References

The following sections provide references related to implementing secure shell.

Related Documents

Related Topic	Document Title
AAA commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Authentication, Authorization, and Accounting Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5
AAA configuration tasks	*Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*, Release 3.5
Host services and applications commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Host Services and Applications Commands on Cisco IOS XR Software* module in the *Cisco IOS XR IP Addresses and Services Command Reference*, Release 3.5

Related Topic	Document Title
IPSec commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*IPSec Network Security Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5
SSH commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Secure Shell Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5

Standards

Standards	Title
Draft-ietf-secsh-userauth-17.txt	*SSH Authentication Protocol*, July 2003
Draft-ietf-secsh-connect-17.txt	*SSH Connection Protocol*, July 2003
Draft-ietf-secsh-architecture-14.txt	*SSH Protocol Architecture*, July 2003
Draft-ietf-secsh-transport-16.txt	*SSH Transport Layer Protocol*, July 2003

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, and support for existing RFCs has not been modified 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

Implementing Secure Socket Layer on Cisco IOS XR Software

The Secure Socket Layer (SSL) protocol and Transport Layer Security (TLS) are application-level protocols that provide for secure communication between a client and server by allowing mutual authentication, the use of hash for integrity, and encryption for privacy. SSL and TLS rely on certificates, public keys, and private keys.

Certificates are similar to digital ID cards. They prove the identity of the server to clients. Certificates are issued by certification authorities (CAs), such as VeriSign or Thawte. Each certificate includes the name of the authority that issued it, the name of the entity to which the certificate was issued, the entity's public key, and time stamps that indicate the certificate's expiration date.

Public and private keys are the ciphers used to encrypt and decrypt information. Although the public key is shared quite freely, the private key is never given out. Each public-private key pair works together: Data encrypted with the public key can be decrypted only with the private key.

This module describes the tasks that you need to implement SSL on your Cisco IOS XR network.

Note For a complete description of the Public Key Infrastructure (PKI) commands used in this chapter, see the *Public Key Infrastructure Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference* publication. For information on SSL commands, see the *Secure Socket Layer Protocol Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference* publication. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing Secure Socket Layer on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.3.0	No modification.
Release 3.4.0	No modification.
Release 3.5.0	No modification.

Contents

- Prerequisites for Implementing Secure Socket Layer, page SC-160
- Information About Implementing Secure Socket Layer, page SC-160
- How to Implement Secure Socket Layer, page SC-161
- Configuration Examples for Implementing Secure Socket Layer, page SC-164
- Additional References, page SC-164

Prerequisites for Implementing Secure Socket Layer

The following prerequisites are required to implement SSL:

- You must be in a user group associated with a task group that includes the proper task IDs for security commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- You must install and activate the Package Installation Envelope (PIE) for the security software.

  For detailed information about optional PIE installation, refer to the *Cisco IOS XR Getting Started Guide*.
- Before you can begin using SSL, you must generate either Rivest, Shamir, and Adelman (RSA) or Digital Signature Algorithm (DSA) key pairs, enroll with a CA, and obtain the CA certificate for the router key.

  For more information on the commands required to perform these tasks, see the **crypto key generate rsa**, **crypto key generate dsa**, **crypto ca enroll**, and **crypto ca authenticate** commands in the *Public Key Infrastructure Commands on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Command Reference*.

Information About Implementing Secure Socket Layer

To implement SSL you need to understand the following concept:

- Purpose of Certification Authorities, page SC-160

Purpose of Certification Authorities

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 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. Internet Key Exchange (IKE), an essential component of IPSec, can use digital signatures to scalable authenticate peer devices before setting up security associations (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.

How to Implement Secure Socket Layer

To configure SSL so that it can be used by any application, such as HTTP server or object request broker (ORB) server, perform the task described in the following section.

- Configuring Secure Socket Layer, page SC-161 (required)

Configuring Secure Socket Layer

This task explains how to configure SSL.

SUMMARY STEPS

1. **crypto key generate rsa** [**usage-keys** | **general-keys**] [*keypair-label*]
2. **configure**
3. **domain ipv4 host** *host-name v4address1* [*v4address2...v4address8*] [**unicast** | **multicast**]
4. **crypto ca trustpoint** *ca-name*
5. **enrollment url** *CA-URL*
6. **end**
   or
   commit
7. **crypto ca authenticate** *ca-name*
8. **crypto ca enroll** *ca-name*
9. **show crypto ca certificates**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**crypto key generate rsa** [**usage-keys** | **general-keys**] [*keypair-label*]
	
	Example:
	`RP/0/RP0/CPU0:router# crypto key generate rsa general-keys`
	`The name for the keys will be: the_default`
	`% You already have keys defined for the_default`
	`Do you really want to replace them? [yes/no]:`
		Generates RSA key pairs.
		• RSA key pairs are used to sign and encrypt Internet Key Exchange (IKE) key management messages and are required before you can obtain a certificate for your router.
		• 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.
		• To generate DSA key pairs, use the **cypto key generate dsa** command in EXEC mode.

Step 2	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**domain ipv4 host** *host-name v4address1* [*v4address2...v4address8*] [**unicast** | **multicast**] **Example:** `RP/0/RP0/CPU0:router(config)# domain ipv4 host ultra5 192.168.7.18`	Defines a static hostname-to-address mapping in the host cache using IPv4. • To define a static hostname-to-address mapping in the host cache using IPv6, use the **domain ipv6 host** *hostname v6address1* [*v6address2...v6address8*] [**unicast** | **multicast**] command.
Step 4	**crypto ca trustpoint** *ca-name* **Example:** `RP/0/RP0/CPU0:router(config)# crypto ca trustpoint myca`	Configures a trusted point with a selected name so that your router can verify certificates issued to peers. • Enters trustpoint configuration mode.

Step 5	**enrollment url** ***CA-URL***
	
	Example:
	`RP/0/RP0/CPU0: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.

	Command or Action	Purpose

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-trustp)# end`
	or
	`RP/0/RP0/CPU0:router(config-trustp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 7	**crypto ca authenticate** *ca-name* **Example:** `RP/0/RP0/CPU0:router# crypto ca authenticate myca`	This command authenticates the CA to your router by obtaining the CA certificate, which contains the public key for the CA. • When prompted, type **y** to accept the certificate.
Step 8	**crypto ca enroll** *ca-name* **Example:** `RP/0/RP0/CPU0:router# crypto ca enroll myca`	Requests certificates for all of your RSA key pairs. • This command causes your router to request as many certificates as there are RSA key pairs, so you need only perform this command once, even if you have special usage RSA key pairs. • This command requires you to create a challenge password that is not saved with the configuration. This password is required if your certificate needs to be revoked, so you must remember this password. • A certificate may be issued immediately or the router sends a certificate request every minute until the enrollment retry period is reached and a timeout occurs. If a timeout occurs, contact your system administrator to get your request approved, and then enter this command again. • Verify that the certificate has been granted by using the **show crypto ca certificates** command.

Step 9	`show crypto ca certificates`
	
	Example:
	`RP/0/RP0/CPU0:router# show crypto ca certificates`
		Displays information about your certificate and the CA certificate.

Configuration Examples for Implementing Secure Socket Layer

This section provides the following configuration example:

- Configuring Secure Socket Layer: Example, page SC-164

Configuring Secure Socket Layer: Example

The following example shows how to generate the RSA keys for the router, configure a trust point, authenticate the CA server, obtain a certificate from the CA for the key, and display information about the certificate:

```
crypto key generate rsa general-keys
commit
configure
    domain ipv4 host xyz-ultra5 10.0.0.5
    crypto ca trustpoint myca
        enrollment url http://xyz-ultra5
        end
crypto ca authenticate myca
crypto ca enroll myca
show crypto ca certificates
```

Additional References

The following sections provide references related to implementing SSL.

Related Documents

Related Topic	Document Title
PKI commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Public Key Infrastructure Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5
SSL commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Secure Socket Layer Protocol Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*, Release 3.5
Certification authority information	*Implementing Certification Authority Interoperability on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*, Release 3.5

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
RFC 2246	*The TLS Protocol, Version 1, T. Dierks, C. Allen. January 1999.*

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

Configuring AAA Services on Cisco IOS XR Software

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

This module describes the tasks that you need to configure AAA services on your Cisco IOS XR network.

Note For a complete description of the AAA commands listed in this module, see the *Authentication, Authorization, and Accounting Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference* publication. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Configuring AAA Services on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.

Release 3.3.0	• Support for the RADIUS Dead-Server Detection feature was added. • To enable interoperability based on Cisco IOS software, tasks must be marked as an optional attribute. • Support was added on Cisco IOS XR to allow 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 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. • A procedure on how to specify the task ID and usergroups by using the CiscoSecure ACS was added. • All references to owner secure domain router (SDR) were replaced with root SDR. • Support was added to prompt the next logged-in user for a new username and password if all the users were deleted. • The predefined task group serviceadmin was added. • An example was added for RADIUS Vendor -Specific Attribute (VSA). • EXEC authorization was added to the "Administrative Access" section.
Release 3.4.0	• The **server-private** command was added to configure RADIUS server groups. • Support for the Per VRF AAA feature was added. • Support for generating interim accounting records was added.
Release 3.5.0	No modification.

Contents

- Prerequisites for Configuring AAA Services, page SC-169
- Restrictions for Configuring AAA Services, page SC-169
- Information About Configuring AAA Services, page SC-169
- How to Configure AAA Services, page SC-183
- Configuration Examples for Configuring AAA Services, page SC-221
- Additional References, page SC-223

Prerequisites for Configuring AAA Services

The following prerequisites are listed:

- You must be in a user group associated with a task group that includes the proper task IDs for security commands.
- 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 Cisco IOS XR 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" 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, page SC-170
- Cisco IOS XR Software Administrative Model, page SC-172
- Password Types, page SC-177
- Task-Based Authorization, page SC-178
- Task IDs for TACACS+ and RADIUS Authenticated Users, page SC-179
- XML Schema for AAA Services, page SC-181
- About RADIUS, page SC-182

User, User Groups, and Task Groups

Cisco IOS XR software user attributes form the basis of the Cisco IOS XR 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) (see the "Task IDs" section)

User Categories

Router users are classified into the following categories:

- Root system user (complete administrative authority)
- Root SDR user (specific secure domain router administrative authority)
- Secure domain router user (specific secure domain router user access)

Root System Users

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

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

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

Root SDR Users

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

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

- Create, delete, and modify secure domain router users and their privileges for the SDR. (See the "Secure Domain Router Users" section.)
- Create, delete, and modify user groups to allow access to the SDR.
- Manage nearly all aspects of the SDR.

A root SDR user cannot deny access to a root system user. (See the "Root System Users" section.)

Secure Domain Router Users

A secure domain router user has restricted access to an SDR as determined by the root-system user or root SDR user. The secure domain router user performs the day-to-day system and network management activities. The tasks that the secure domain router user is allowed to perform are determined by the task IDs associated with the user groups to which the secure domain router user belongs. (See the "User Groups" section.)

User Groups

Cisco IOS XR software allows the system administrator to configure groups of users and the job characteristics that are common in groups of users. Groups must be explicitly assigned to users. Users are not assigned to groups by default. A user can be assigned to more than one group.

A *user group* defines a collection of users that share a set of attributes, such as access privileges. Each user may be associated with one or more user groups. User groups have the following attributes:

- List of task groups that define the authorization for the users. All tasks, except cisco-support, are permitted by default for root system users. (See the "Root System Users" section.)
- Each user task can be assigned read, write, execute, or debug permission.

Predefined User Groups

The Cisco IOS XR 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.
- netadmin: Has the ability to control and monitor all system and network parameters.
- operator: A demonstration group with basic privileges.
- root-lr: Has the ability to control and monitor the specific secure domain router.
- root-system: Has the ability to control and monitor the entire system.
- 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).

The user group root-system has root system users as the only members. (See the "Root System Users" section.) The root-system user group has predefined authorization; that is, it has the complete responsibility for root-system user-managed resources and certain responsibilities in other SDRs.

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, and change in group B affects group A, even if the group is not re-inherited explicitly.

Task Groups

A task group is defined by a collection of task IDs. Task groups contain task ID lists for each class of action.

Each user group is associated with a set of task groups applicable to the users in that group. A user's task permissions are derived from the task groups associated with the user groups to which that user belongs.

Predefined Task Groups

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

- cisco-support: Cisco support personnel tasks
- netadmin: Network administrator tasks
- operator: Operator day-to-day tasks (for demonstration purposes)
- root-lr: Secure domain router administrator tasks
- root-system: System-wide 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" 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.

Cisco IOS XR Software 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.

The root-system user has the highest level of responsibility for the router. This user provisions secure domain routers and creates root SDR users. After being created, root SDR users take most of the responsibilities from the root-system user for the SDR. Root SDR users in turn can create secure domain router users. Root-system users and root SDR users have fixed permissions (task IDs) that cannot be changed by users.

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. A lockout of all root-system users is a serious issue that requires a system reload to recover the password.

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

  The **none** option for authentication is not supported in Cisco IOS XR software. Cisco IOS XR user access is more secure than Cisco IOS software, and there is no way that a user can access the system without a valid username and password.

- Removing the flash card from disk0:, or a disk corruption, may deny auxiliary port authentication, which can affect certain system debugging abilities. However, if the console is available, the system is still accessible.
- Configuring command authorization or EXEC 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 one or both of the following:

- Before turning on TACACS+ command authorization or EXEC 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 EXEC 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.

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 SDR in which it is stored, and the defined users or groups are not visible to other secure domain routers in the same Cisco CRS-1.

Unlike releases earlier than Release 3.3.0, you are able to delete the last remaining user from the local database. If all users are deleted and 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 section.

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 and auxiliary 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)

Server Grouping

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

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 EXEC 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 of Nonowner Secure Domain Router User

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

Authentication of Owner Secure Domain Router User

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

Authentication of Secure Domain Router User

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

Authentication Flow of Control

AAA performs authentication according to the following process:

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

- If the user has been configured as a member of a root-system user group, then AAA authenticates the user as a root-system 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 a root-system user group or 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 EXEC and external API clients can use this feature to optimize their operations. EXEC 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.

Ksh 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) 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:/ ...
```

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 decryptable password, such as login. For applications such as PPP, the two-way encrypted password is used even if a secret is present.

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 Cisco IOS XR 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

Every router control, configuration, or monitoring operation (CLI or XML API) is associated with a particular set of task IDs. A given CLI command or API invocation is associated with at least one or more task IDs. 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 Table 6.

Table 6 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, read/write) 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 none specified, the command is used without previous user association to a task ID and operation. In addition, users do not need to be associated to task IDs to use ROM monitor commands.

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

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

Whereas, in administration EXEC mode, a user needs to be associated to both admin and system (read) task IDs and operations, as shown in the following example:

```
RP/0/RP0/CPU0:router# admin
RP/0/RP0/CPU0:router(admin)# show redundancy
```

Task IDs for TACACS+ and RADIUS Authenticated Users

Cisco IOS XR 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.

  See the "Task Maps" section for more details.

- Specify the privilege level in the configuration file of the external TACACS+ and RADIUS servers.

  See the "Privilege Level Mapping" section for more details.

- 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 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 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 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:

Step 1 Enter your username and password.

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

Step 3 Select the group that you want to update from the Group drop-down list.

Step 4 Click the **Edit Settings** button to display the Group Settings window.

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:

```
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:

```
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 EXEC mode to display all the tasks user1 can perform. For example:

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

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

```
Task:          ext-access  :READ             EXECUTE
Task:             logging  :READ
```

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

```
Username:user2
Password:
RP/0/RP0/CPU0: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 15, the root-system user group is used; 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.

With the RADIUS server, task IDs are defined using the Cisco-AVPair, as shown in the following example:

```
user = sampleuser2{
    member = bar
    Cisco-AVPair = "shell:tasks=#root-system,#cisco-support"{
        Cisco-AVPair = "shell:priv-lvl=10"
    }
}
```

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

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. 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 wish to 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.

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.
   b. REJECT—The user is not authenticated and is prompted to reenter the username and password, or access is denied.
   c. CHALLENGE—A challenge is issued by the RADIUS server. The challenge collects additional data from the user.
   d. 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 that is used for EXEC 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 EXEC services.
- Connection parameters, including the host or client IP address, access list, and user timeouts.

How to Configure AAA Services

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

- Configuring Task Groups, page SC-184 (required)
- Configuring User Groups, page SC-186 (required)
- Configuring Users, page SC-188 (required)
- Configuring Router to RADIUS Server Communication, page SC-190 (required)
- Configuring RADIUS Dead-Server Detection, page SC-194 (required)
- Configuring Per VRF AAA, page SC-196 (required)
- Configuring a TACACS+ Server, page SC-198 (required)
- Configuring RADIUS Server Groups, page SC-201 (required)
- Configuring TACACS+ Server Groups, page SC-203 (required)
- Configuring AAA Method Lists, page SC-204 (required)

- Applying Method Lists for Applications, page SC-216 (required)
- Configuring Login Parameters, page SC-220 (required)

Configuring Task Groups

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 selected from the Cisco CRS-1 set of available task IDs. The first configuration task in setting up the Cisco CRS-1 authorization scheme is to configure the task groups, followed by user groups, followed by individual users.

Task Group Configuration

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

The **inherit taskgroup** command may be used to derive permissions from another group. Cyclic references are detected and rejected. It is not possible to inherit from the root-system and owner-sdr predefined groups.

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 will result in an error.

Prerequisites

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

Restrictions

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

SUMMARY STEPS

1. **configure**
2. **taskgroup** *taskgroup-name*
3. **description** *string*
4. **inherit taskgroup** *taskgroup-name*
5. **task** {**read** | **write** | **execute** | **debug**} *taskid-name*
6. Repeat Step 5 for each task ID to be associated with the task group named in Step 2.
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**taskgroup** *taskgroup-name* **Example:** `RP/0/RP0/CPU0:router(config)# taskgroup beta`	Creates a name for a particular task group and enters task group configuration submode. • Specific task groups can be removed from the system by specifying the **no** form of the **taskgroup** command.
Step 3	**description** *string* **Example:** `RP/0/RP0/CPU0:router(config-tg)# description this is a sample task group description`	(Optional) Creates a description of the task group named in Step 2.
Step 4	**inherit taskgroup** *taskgroup-name* **Example:** `RP/0/RP0/CPU0:router(config-tg)# inherit taskgroup sysadmin`	(Optional) Derives permissions from another task group and assigns them to the task group named in Step 2. • Cyclic references are detected and rejected. • To explicitly define permissions for the task group named in Step 2, omit Step 4 and go to Step 5.
Step 5	**task** {**read** | **write** | **execute** | **debug**} *taskid-name* **Example:** `RP/0/RP0/CPU0:router(config-tg)# task read bgp`	Specifies a task ID to be associated with the task group named in Step 2. • Assigns **read** permission for any CLI or API invocations associated with that task ID and performed by a member of the task group. • Specific task IDs can be removed from a task group by specifying the **no** prefix for the **task** command.

	Command or Action	Purpose
Step 6	Repeat Step 5 for each task ID to be associated with the task group named in Step 2.	—

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-tg)# end`
	or
	`RP/0/RP0/CPU0:router(config-tg)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

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.

Configuring 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.

Use the **inherit usergroup** command to inherit permissions from other user groups. The user group is inherited by the parent group and form a union of all task IDs specified in those groups. Cyclic inclusions are detected and rejected.

Restrictions

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

SUMMARY STEPS

1. **configure**
2. **usergroup** *usergroup-name*

3. **description** *string*
4. **inherit usergroup** *usergroup-name*
5. **taskgroup** *taskgroup-name*
6. Repeat Step 5 for each task group to be associated with the user group named in Step 2.
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**usergroup** *usergroup-name* **Example:** `RP/0/RP0/CPU0:router(config)# usergroup beta`	Creates a name for a particular user group and enters user group configuration submode. • Specific user groups can be removed from the system by specifying the **no** form of the **usergroup** command.
Step 3	**description** *string* **Example:** `RP/0/RP0/CPU0:router(config-ug)# description this is a sample user group description`	(Optional) Creates a description of the user group named in Step 2.
Step 4	**inherit usergroup** *usergroup-name* **Example:** `RP/0/RP0/CPU0:router(config-ug)# inherit usergroup sales`	Derives permissions from another user group and assigns them to the user group named in Step 2. • Cyclic inclusions is detected and rejected. Permissions may not be inherited from predefined user groups, such as root-system and owner-sdr. • To explicitly define permissions for the user group named in Step 2, omit Step 4 and go to Step 5.
Step 5	**taskgroup** *taskgroup-name* **Example:** `RP/0/RP0/CPU0:router(config-ug)# taskgroup beta`	Associates the user group named in Step 2 with the task group named in this step. • The user group takes on the configuration attributes (task ID list and permissions) already defined for the entered task group.

	Command or Action	Purpose
Step 6	Repeat Step 5 for each task group to be associated with the user group named in Step 2.	—

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ug)# end`
	or
	`RP/0/RP0/CPU0:router(config-ug)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After completing configuration of a full set of user groups, configure individual users as described in the "Configuring Users" section.

Configuring 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.

SUMMARY STEPS

1. **configure**
2. **username** *user-name*
3. **password** {**0** | **7**} *password*
   or
   secret {**0** | **5**} *secret*
4. **group** *group-name*
5. Repeat Step 4 for each user group to be associated with the user specified in Step 2.

6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**username** *user-name* **Example:** `RP/0/RP0/CPU0:router(config)# username user1`	Creates a name for a new user (or identifies a current user) and enters username configuration submode. • The *user-name* argument can be only one word. Spaces and quotation marks are not allowed.

Step 3	**password** {**0** | **7**} *password*
	or
	secret {**0** | **5**} *secret*
	
	Example:
	`RP/0/RP0/CPU0:router(config-un)# password 0 pwd1`
	or
	`RP/0/RP0/CPU0:router(config-un)# secret 0 sec1`
		Specifies a password for the user named in Step 2.
		• Use the **secret** command to create a secure login password for the user names specified in Step 2.
		• Entering **0** following the **password** command specifies that an unencrypted (clear-text) password follows. Entering **7** following the **password** command specifies that an encrypted password follows.
		• Entering **0** following the **secret** command specifies that a secure unencrypted (clear-text) password follows. Entering **5** following the **secret** command specifies that a secure encrypted password follows.
		• Type **0** is the default for the **password** and **secret** commands.

Step 4	**group** *group-name* **Example:** `RP/0/RP0/CPU0:router(config-un)# group sysadmin`	Assigns the user named in Step 2 to a user group that has already been defined through the **usergroup** command. • The user takes on all attributes of the user group, as defined by that user group's association to various task groups. • Each user must be assigned to at least one user group. A user may belong to multiple user groups.

	Command or Action	Purpose
Step 5	Repeat Step 4 for each user group to be associated with the user specified in Step 2.	—

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-un)# end`
	or
	`RP/0/RP0/CPU0:router(config-un)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After completing configuration of a full set of users, configure router to use the RADIUS server communication or TACACS+ servers (See the "Configuring Router to RADIUS Server Communication" or "Configuring a TACACS+ Server" section.)

Configuring 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.

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.

SUMMARY STEPS

1. **configure**
2. **radius-server host** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*]
3. **radius-server retransmit** *retries*
4. **radius-server timeout** *seconds*
5. **radius-server key** {**0** *clear-text-key* | **7** *encrypted-key* | *clear-text-key*}
6. **radius source-interface** *type instance* [**vrf** *vrf-id*]
7. Repeat Step 2 through Step 6 for each external server to be configured.
8. **end**
   or
   commit
9. **show radius**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**radius-server host** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*] **Example:** `RP/0/RP0/CPU0:router(config)# radius-server host host1`	Specifies the hostname or IP address of the remote RADIUS server host. • Use the **auth-port** *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for authentication. • Use the **acct-port** *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for accounting. • To configure the network access server to recognize more than one host entry associated with a single IP address, simply repeat this command as many times as necessary, making sure that each UDP port number is different. Set the timeout, retransmit, and encryption key values to use with the specific RADIUS host. • If no timeout is set, the global value is used; otherwise, enter a value in the range 1 to 1000. If no retransmit value is set, the global value is used; otherwise enter a value in the range 1 to 100. If no key string is specified, the global value is used. **Note** The key is a text string that must match the encryption key used on the RADIUS server. Always configure the key as the last item in the **radius-server host** command syntax because the leading spaces are ignored, but spaces within and at the end of the key are used. If you use spaces in your key, do not enclose the key in quotation marks unless the quotation marks themselves are part of the key.
Step 3	**radius-server retransmit** *retries* **Example:** `RP/0/RP0/CPU0:router(config)# radius-server retransmit 5`	Specifies the number of times the Cisco IOS XR software searches the list of RADIUS server hosts before giving up. • In the example, the number of retransmission attempts is set to 5.
Step 4	**radius-server timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config)# radius-server timeout 10`	Sets the number of seconds a router waits for a server host to reply before timing out. • In the example, the interval timer is set to 10 seconds.

	Command or Action	Purpose
Step 5	**radius-server key {0** *clear-text-key* **| 7** *encrypted-key* **|** *clear-text-key***}** **Example:** `RP/0/RP0/CPU0:router(config)# radius-server key 0 samplekey`	Sets the authentication and encryption key for all RADIUS communications between the router and the RADIUS daemon.
Step 6	**radius source-interface** *type instance* [**vrf** *vrf-id*] **Example:** `RP/0/RP0/CPU0:router(config)# radius source-interface POS 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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**show radius** **Example:** `RP/0/RP0/CPU0:router# show radius`	(Optional) Displays information about the RADIUS servers that are configured in the system.

What to Do Next

After configuring router to RADIUS server communication, configure RADIUS server groups. (See the "Configuring RADIUS Server Groups" section.)

Configuring RADIUS Dead-Server Detection

This task configures the RADIUS Dead-Server Detection feature.

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

SUMMARY STEPS

1. **configure**
2. **radius-server deadtime** *minutes*
3. **radius-server dead-criteria time** *seconds*
4. **radius-server dead-criteria tries** *tries*
5. **end**
   or
   commit
6. **show radius dead-criteria host** *ip-addr* [**auth-port** *auth-port*] [**acct-port** *acct-port*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**radius-server deadtime** *minutes* **Example:** `RP/0/RP0/CPU0:router(config)# radius-server deadtime 5`	Improves RADIUS response times when some servers might be unavailable and causes the unavailable servers to be skipped immediately.
Step 3	**radius-server dead-criteria time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config)# radius-server dead-criteria time 5`	Establishes the time for the dead-criteria conditions for a RADIUS server to be marked as dead.
Step 4	**radius-server dead-criteria tries** *tries* **Example:** `RP/0/RP0/CPU0:router(config)# radius-server dead-criteria tries 4`	Establishes the number of tries for the dead-criteria conditions for a RADIUS server to be marked as dead.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**show radius dead-criteria host** *ip-addr* [**auth-port** *auth-port*] [**acct-port** *acct-port*] **Example:** `RP/0/RP0/CPU0:router# show radius dead-criteria host 172.19.192.80`	(Optional) Displays dead-server-detection information that has been requested for a RADIUS server at the specified IP address.

Configuring 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 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 TACACS+ specification, and “sep” is “=” for mandatory attributes and “*” for optional attributes. This definition allows the full set of features available for TACACS+ authorization to also be used for RADIUS.

Table 7 describes the VSAs that are now supported for Per VRF AAA.

Table 7 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.		
rad-serv	string	Indicates the IP address, 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.

SUMMARY STEPS

1. **configure**
2. **aaa group server radius** *group-name*

3. **server-private** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*]
4. **vrf** *vrf-name*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**aaa group server radius** *group-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# aaa group server radius radgroup1`
	`RP/0/RP0/CPU0: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*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*]
	
	Example:
	`RP/0/RP0/CPU0:router(config-sg-radius)# server-private 10.1.1.1 timeout 5`
	`RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0: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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-sg-radius)# end`
	or
	`RP/0/RP0/CPU0:router(config-sg-radius)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a 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 the TACACS+ servers. The timeout specifies how long the AAA server waits to receive a response from the TACACS+ server. The key specifies an authentication and encryption key shared between the AAA server and the TACACS+ server.

SUMMARY STEPS

1. **configure**
2. **tacacs-server host** *host-name* **port** *port-number*
3. **tacacs-server host** *host-name* **timeout** *seconds*
4. **tacacs-server host** *host-name* **key** [**0** | **7**] *auth-key*
5. **tacacs-server host** *host-name* **single-connection**
6. **tacacs source-interface** *type instance*

7. Repeat Step 2 through Step 5 for each external server to be configured.
8. **end**
   or
   commit
9. **show tacacs**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**tacacs-server host** *host-name* **port** *port-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# tacacs-server host 209.165.200.226 port 51`
	`RP/0/RP0/CPU0:router(config-tacacs-host)#`
		Specifies a TACACS+ host server and optionally specifies a server port number.
		• This option overrides the default, port 49. Valid port numbers range from 1 to 65535.

Step 3	**tacacs-server host** *host-name* **timeout** *seconds*
	
	Example:
	`RP/0/RP0/CPU0:router(config-tacacs-host)# tacacs-server host 209.165.200.226 timeout 30`
	`RP/0/RP0/CPU0:router(config)#`
		Specifies a TACACS+ host server and optionally specifies a timeout value that sets the length of time the AAA server will wait to receive a response from the TACACS+ server.
		• This option overrides the global timeout value set with the **tacacs-server timeout** command for this server only. The timeout value is expressed as an integer in terms of timeout interval seconds. The valid timeout range is from 1 to 1000 seconds.

Step 4	**tacacs-server host** *host-name* **key** [**0** | **7**] *auth-key* **Example:** `RP/0/RP0/CPU0:router(config)# tacacs-server host 209.165.200.226 key 0 a_secret`	Specifies a TACACS+ host server and optionally specifies an authentication and encryption key shared between the AAA server and the TACACS+ server. • The TACACS+ packets are encrypted using this key. This key must match the key used by TACACS+ daemon. Specifying this key overrides the global key set by the **tacacs-server key** command for this server only. • (Optional) Entering **0** indicates that an unencrypted (clear-text) key will follow. • (Optional) Entering **7** indicates that an encrypted key will follow. • The *auth-key* argument specifies the encrypted or unencrypted key to be shared between the AAA server and the TACACS+ server.

	Command or Action	Purpose
Step 5	**tacacs-server host** *host-name* **single-connection** **Example:** `RP/0/RP0/CPU0:router(config)# tacacs-server host 209.165.200.226 single-connection`	Prompts the router to multiplex all TACACS+ requests to this server over a single TCP connection. By default, a separate connection is used for each session.
Step 6	**tacacs source-interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config)# tacacs source-interface POS 0/4/0/0`	(Optional) Specifies the source IP address of a selected interface for all outgoing TACACS+ packets. • The specified interface or subinterface must have an IP address associated with it. If the specified interface or subinterface does not have an IP address or is in the down state, then TACACS+ reverts to the default. To avoid this, add an IP address to the interface or subinterface or bring the interface to the up state.
Step 7	Repeat Step 2 through Step 5 for each external server to be configured.	—

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**show tacacs** **Example:** `RP/0/RP0/CPU0:router# show tacacs`	(Optional) Displays information about the TACACS servers that are configured in the system.

What to Do Next

After configuring TACACS+ servers, configure TACACS+ server groups. (See the "Configuring TACACS+ Server Groups" section.)

Configuring 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). (See the "Method Lists" section.)

Prerequisites

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

SUMMARY STEPS

1. **configure**
2. **aaa group server radius** *group-name*
3. **server** {*host-name* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*]
4. Repeat Step 3 for every external server to be added to the server group named in Step 2.
5. **server-private** {*hostname* | *ip-address*} [**auth-port** *port-number*] **[acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*]
6. **deadtime** *minutes*
7. **end**
   or
   commit
8. **show radius server-groups** [*group-name* [**detail**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**aaa group server radius** *group-name* **Example:** `RP/0/RP0/CPU0:router(config)# aaa group server radius radgroup1`	Groups different server hosts into distinct lists and enters the server group configuration mode.
Step 3	**server** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*] **Example:** `RP/0/RP0/CPU0:router(config-sg-radius)# server 192.168.20.0`	Specifies the hostname or IP address of an external RADIUS server. • After the server group is configured, it can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

	Command or Action	Purpose
Step 4	Repeat Step 3 for every external server to be added to the server group named in Step 2.	—
Step 5	**server-private** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*] [**timeout** *seconds*] [**retransmit** *retries*] [**key** *string*] **Example:** `RP/0/RP0/CPU0:router(config-sg-radius)# server-private 10.10.130.2 auth-port 1600 acct-port 1666 key code`	Configures the IP address of the private RADIUS server for the group server. **Note** If private server parameters are not specified, global configurations are used. If global configurations are not specified, default values are used.
Step 6	**deadtime** *minutes* **Example:** `RP/0/RP0/CPU0:router(config-sg-radius)# deadtime 1`	Configures the deadtime value at the RADIUS server group level. • The *minutes* argument specifies the length of time, in minutes, for which a RADIUS server is skipped over by transaction requests, up to a maximum of 1440 (24 hours). The range is from 1 to 1440. The example specifies a one-minute deadtime for RADIUS server group radgroup1 when it has failed to respond to authentication requests for the **deadtime** command **Note** You can configure the group-level deadtime after the group is created.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-sg-radius)# end`
	or
	`RP/0/RP0/CPU0:router(config-sg-radius)# commit`
		Saves configuration changes.
		
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show radius server-groups** [*group-name* [**detail**]] **Example:** `RP/0/RP0/CPU0:router# show radius server-groups`	(Optional) Displays information about each RADIUS server group that is configured in the system.

What to Do Next

After configuring RADIUS server groups, define method lists by configuring authentication, authorization, and accounting. (See the "Configuring AAA Method Lists" section.)

Configuring TACACS+ Server Groups

This task configures TACACS+ server groups.

The user 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). (See the "Method Lists" section.)

Prerequisites

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

SUMMARY STEPS

1. **configure**
2. **aaa group server tacacs+** *group-name*
3. **server** {*host-name* | *ip-address*}
4. Repeat Step 3 for every external server to be added to the server group named in Step 2.
5. **end**
   or
   commit
6. **show tacacs server-groups**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**aaa group server tacacs+** *group-name* **Example:** `RP/0/RP0/CPU0:router(config)# aaa group server tacacs+ tacgroup1`	Groups different server hosts into distinct lists and enters the server group configuration mode.
Step 3	**server** {*hostname* | *ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-sg-tacacs+)# server 192.168.100.0`	Specifies the hostname or IP address of an external TACACS+ server. • When configured, this group can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

	Command or Action	Purpose
Step 4	Repeat Step 3 for every external server to be added to the server group named in Step 2.	—

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-sg-tacacs+)# end`
	or
	`RP/0/RP0/CPU0:router(config-sg-tacacs+)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**show tacacs server-groups** **Example:** `RP/0/RP0/CPU0:router# show tacacs server-groups`	(Optional) Displays information about each TACACS+ server group that is configured in the system.

What to Do Next

After configuring TACACS+ server groups, define method lists by configuring authentication, authorization, and accounting. (See the "Configuring AAA Method Lists" section.)

Configuring 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 and aux 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, page SC-205 (required)
- Configuring Authorization Method Lists, page SC-207 (required)
- Configuring Accounting Method Lists, page SC-210 (required)

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 and aux 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.

Creation of a Series of Authentication Methods

Use the **aaa authentication** command to create a series of authentication methods, or method list. A method list is a named list describing the authentication methods to be used (such as RADIUS or TACACS+), in sequence. The method will be one of the following:

- group radius—Use a server group or RADIUS servers for authentication
- group tacacs+—Use a server group or TACACS+ servers for authentication
- local—Use the local username or password database for authentication
- line—Use the line password or user group for authentication

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.

Restrictions

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.

Note 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.

SUMMARY STEPS

1. **configure**
2. **aaa authentication** {**login** | **ppp**} {**default** | *list-name* | **remote**} *method-list*

3. **end**
   or
   commit
4. Repeat Step 1 through Step 3 for every authentication method list to be configured.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**aaa authentication** {**login** | **ppp**} {**default** | *list-name* | **remote**} *method-list* **Example:** `RP/0/RP0/CPU0:router(config)# aaa authentication login default group tacacs+`	Creates a series of authentication methods, or a method list. • Using the **login** keyword sets authentication for login. Using the **ppp** keyword sets authentication for Point-to-Point Protocol. • Entering the **default** keyword causes the listed authentication methods that follow this keyword to be the default list of methods for authentication. • Entering a *list-name* character string identifies the authentication method list. • Entering the **remote** keyword causes the listed authentication methods that follow this keyword to be the default list of methods for administrative authentication on a remote nonowner SDR. **Note** The remote keyword is available only on the admin plane. • 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.

	Command or Action (continued)	Purpose (continued)

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

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

What to Do Next

After configuring authentication method lists, configure authorization method lists. (See the "Configuring Authorization Method Lists" section).

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 Cisco IOS XR software uses the first method listed to authorize users for specific network services; if that method fails to respond, the Cisco IOS XR 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 Cisco IOS XR 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. The Cisco IOS XR software supports three types of AAA authorization:

- Command authorization: Applies to the EXEC mode commands a user issues. Command authorization attempts authorization for all EXEC mode commands.
- EXEC authorization: Applies authorization for starting an EXEC session.
- Network authorization: Applies authorization for network services such as Internet Key Exchange (IKE).

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 will be 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.

Creation of a Series of Authorization Methods

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.

The Cisco IOS XR software supports the following methods for authorization:

- none—The router does not request authorization information; authorization is not performed over this line or interface.
- local—Uses local database for authorization.
- group tacacs+—Uses the list of all configured TACACS+ servers for authorization.
- group radius—Uses the list of all configured RADIUS servers for authorization.

SUMMARY STEPS

1. **configure**
2. **aaa authorization** {**commands** | **exec** | **network**} {**default** | *list-name*} {**none** | **local** | **group** {**tacacs+** | **radius** | *group-name*}}
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**aaa authorization** {**commands** | **exec** | **network**} {**default** | *list-name*} {**none** | **local** | **group** {**tacacs+** | **radius** | *group-name*}} **Example:** `RP/0/RP0/CPU0:router(config)# aaa authorization commands listname1 group tacacs+`	Creates a series of authorization methods, or a method list. • The **commands** keyword configures authorization for all EXEC shell commands. Command authorization applies to the EXEC mode commands issued by a user. Command authorization attempts authorization for all EXEC mode commands. • The **exec** keyword configures authorization for an interactive (EXEC) 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.

	Command or Action (continued)	Purpose (continued)
		– **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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After configuring authorization method lists, configure accounting method lists. (See the "Configuring Accounting Method Lists" section.)

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 are 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.

Creation of a 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.

The Cisco IOS XR software supports the following methods for accounting:

- none—Accounting is not performed over this line or interface.
- group tacacs+—Use the list of all configured TACACS+ servers for accounting.
- group radius—Use the list of all configured RADIUS servers for accounting.

SUMMARY STEPS

1. **configure**
2. **aaa accounting** {**commands** | **exec** | **network**} {**default** | *list-name*} {**start-stop** | **stop-only**} {**none** | *method*}
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**aaa accounting** {**commands** | **exec** | **network**} {**default** | *list-name*} {**start-stop** | **stop-only**} {none | method} **Example:** `RP/0/RP0/CPU0:router(config)# aaa accounting commands default stop-only group tacacs+`	Creates a series of accounting methods, or a method list. • The **commands** keyword enables accounting for EXEC shell commands. • The **exec** keyword enables accounting for an interactive (EXEC) session. • The **network** keyword enables accounting for all network-related service requests, such as Internet Key Exchange (IKE) and 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.

	Command or Action (continued)	Purpose (continued)
		• 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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After configuring method lists, apply those method lists. (See the "Applying Method Lists for Applications" section.)

Generating 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, Cisco IOS XR 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.

SUMMARY STEPS

1. **configure**
2. **aaa accounting update** {**newinfo** | **periodic** *minutes*}
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action (continued)	Purpose (continued)
Step 2	**aaa accounting update** {**newinfo** | **periodic** *minutes*} **Example:** `RP/0/RP0/CPU0:router(config)# aaa accounting 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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

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, auxiliary, and so on). Applying method lists is accomplished by enabling AAA authorization and accounting.

This section contains the following procedures:

- Enabling AAA Authorization, page SC-216 (required)
- Enabling Accounting Services, page SC-217 (required)

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.

SUMMARY STEPS

1. **configure**
2. **line** {**aux** | **console** | **default** | **template** *template-name*}
3. **authorization** {**commands** | **exec**} {**default** | *list-name*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**line** {**aux** | **console** | **default** | **template** *template-name*} **Example:** `RP/0/RP0/CPU0:router(config)# line console`	Enters line template configuration mode. • For more information on configuring line templates, see *Implementing Physical and Virtual Terminals on Cisco IOS XR Software*.

	Command or Action (continued)	Purpose (continued)
Step 3	**authorization** {**commands** | **exec**} {**default** | *list-name*} **Example:** `RP/0/RP0/CPU0:router(config-line)# authorization commands listname5`	Enables AAA authorization for a specific line or group of lines. • The **commands** keyword enables authorization on the selected lines for all commands. • The **exec** keyword enables authorization for an interactive (EXEC) 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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-line)# end`
	or
	`RP/0/RP0/CPU0:router(config-line)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After applying authorization method lists by enabling AAA authorization, apply accounting method lists by enabling AAA accounting. (See the "Enabling Accounting Services" section.)

Enabling Accounting Services

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

SUMMARY STEPS

1. **configure**
2. **line {aux | console | default | template** *template-name*}
3. **accounting** {**commands** | **exec**} {**default** | *list-name*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**line {aux | console | default | template** ***template-name*****}** **Example:** `RP/0/RP0/CPU0:router(config)# line console`	Enters line template configuration mode. • For more information on configuring line templates, see *Implementing Physical and Virtual Terminals on Cisco IOS XR Software*.
Step 3	**accounting** {**commands** | **exec**} {**default** | *list-name*} **Example:** `RP/0/RP0/CPU0:router(config-line)# accounting commands listname7`	Enables AAA accounting for a specific line or group of lines. • The **commands** keyword enables accounting on the selected lines for all EXEC shell commands. • The **exec** keyword enables accounting for an interactive (EXEC) 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	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-line)# end`
	or
	`RP/0/RP0/CPU0:router(config-line)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

What to Do Next

After applying accounting method lists by enabling AAA accounting services, configure login parameters. (See the "Configuring Login Parameters" section.)

Configuring Login Parameters

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

SUMMARY STEPS

1. **configure**
2. **line template** *template-name*
3. **timeout login response** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**line template** *template-name* **Example:** `RP/0/RP0/CPU0:router(config)# line template alpha`	Specifies a line to configure and enters line template configuration mode.

	Command or Action (continued)	Purpose (continued)
Step 3	**timeout login response** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-line)# timeout login response 20`	Sets the interval that the server waits for reply to a login. • The *seconds* argument specifies the timeout interval (in seconds) from 0 to 300. The default is 30 seconds. • The example shows how to change the interval timer to 20 seconds.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-line)# end`
	or
	`RP/0/RP0/CPU0:router(config-line)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Configuring AAA Services

This section provides the following configuration examples:

• Configuring AAA Services: Example, page SC-221

Configuring AAA Services: Example

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-system user. Configure similar settings for username user2.

```
   username user1
```

```
secret lab
group root-system
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 1.1.1.1 port 1 key abc
tacacs-server host 2.2.2.2 port 2 key def
tacacs-server host 3.3.3.3 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 3.3.3.3
exit
```

Additional References

The following sections provide references related to configuring AAA services.

Related Documents

Related Topic	Document Title
AAA services commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Authentication, Authorization, and Accounting Commands on Cisco IOS XR Software* module in *Cisco IOS XR System Security Command Reference*, Release 3.5

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, and support for existing RFCs has not been modified 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

Configuring Software Authentication Manager on Cisco IOS XR Software

Software Authentication Manager (SAM) is a component of the Cisco IOS XR software operating system that ensures that software being installed on the router is safe, and that the software does not run if its integrity has been compromised.

For SAM to verify software during installation, the software to be installed must be in a PIE format. PIEs are digitally signed and SAM verifies the digital signature before allowing bits from that PIE to reside on the router. Each time an installed piece of software is run, SAM ensures that the integrity of the software has not been compromised since it was installed. SAM also verifies that software preinstalled on a flash card has not been tampered with while in transit.

When the initial image or a software package update is loaded on the router, SAM verifies the validity of the image by checking the expiration date of the certificate used to sign the image. If an error message is displayed indicating that your certificate has expired, check the system clock and verify that it is accurate. If the system clock is not set correctly, the system does not function properly. For information on setting the system clock, see the **clock set** command in the *Clock Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Management Command Reference*.

For information on SAM commands, see the *Software Authentication Manager Commands on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Command Reference*.

Implementing Management Plane Protection on Cisco IOS XR Software

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. The following other benefits are described:

- Improved performance for data packets on nonmanagement interfaces.
- Support for network scalability.
- Need for fewer access control lists (ACLs) to restrict access to a device, and prevention of management packet floods on switching and routing interfaces from reaching the CPU.

For information on MPP commands, see the *Management Plane Protection Commands on Cisco IOS XR Software* module in *Cisco IOS XR System Security Command Reference*.

Feature History for Implementing Management Plane Protection on Cisco IOS XR Software

Release	Modification
Release 3.5.0	This feature was introduced on the Cisco CRS-1 and Cisco XR 12000 Series Router.

Contents

- Restrictions for Implementing Management Plane Protection, page SC-228
- Information About Implementing Management Plane Protection, page SC-228
- How to Configure a Device for Management Plane Protection, page SC-229
- Configuration Examples for Implementing Management Plane Protection, page SC-232
- Additional References, page SC-233

Restrictions for Implementing Management Plane Protection

Out-of-band configurations, which configure an interface to allow only management traffic, are not supported for this release.

Information About Implementing Management Plane Protection

Before you enable the Management Plane Protection feature, you should understand the following concepts:

- Inband Management Interface, page SC-228
- Control Plane Protection Overview, page SC-228
- Management Plane, page SC-228
- Management Plane Protection Feature, page SC-229
- Benefits of the Management Plane Protection Feature, page SC-229

Inband Management Interface

An *inband management interface* is a Cisco IOS XR 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*.

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.

Control Plane Policing (CoPP) is a Cisco IOS XR control-plane feature that offers rate limiting of all control-plane traffic. CoPP allows you to configure a quality of service (QoS) filter that manages the traffic flow of control plane packets. This QoS filter helps to protect the control plane of Cisco IOS XR routers and switches against denial-of-service (DoS) attacks and helps to maintain packet forwarding and protocol states during an attack or during heavy traffic loads.

Control Plane Protection is a framework that encompasses all policing and protection features in the control plane. The Control Plane Protection feature extends the policing functionality of the CoPP feature by allowing finer policing granularity. Control Plane Protection also includes a traffic classifier, which intercepts control-plane traffic and classifies it in control-plane categories. Management Plane Protection operates within the Control Plane Protection 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 protocol, which is used for the MPP feature, is disabled by default. When a protocol is enabled, the only default management interfaces can be the RP and standby route processor (SRP) Ethernet interfaces that allow only management traffic. You must configure other interfaces by using the MPP CLI as management interfaces. The feature does provide default management interfaces, such as RP and SRP Ethernet interfaces, which are out-of-band interfaces that allow only management traffic. Using a single CLI command, you can configure, modify, or delete a management interface. When you configure a management interface, no interfaces except that management interface accept network management packets destined to the device.

Following are the management protocols that the MPP feature supports. These management protocols are also the only protocols affected when MPP is enabled.

- SSH, v1 and v2
- 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 nonmanagement interfaces.
- Support for network scalability.
- Simplifies the task of using per-interface ACLs to restrict management access to the device.
- Fewer access control lists (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 task:

- Configuring a Device for Management Plane Protection, page SC-230

Configuring a Device for Management Plane Protection

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 in which SSH is allowed to access the router only through the POS 0/5/0/0 interface.

SUMMARY STEPS

1. **configure**
2. **control-plane**
3. **management-plane**
4. **inband**
5. **interface** {*type instance* | **all**}
6. **allow** {*protocol* | **all**}
7. **end**
   or
   commit
8. **show mgmt-plane** [**interface** {*type instance*}]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**control-plane**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# control-plane`
	`RP/0/RP0/CPU0:router(config-ctrl)#`
		Enters the control plane configuration mode.

Step 3	**management-plane**
	
	Example:
	`RP/0/RP0/CPU0:router(config-ctrl)# management-plane`
	`RP/0/RP0/CPU0:router(config-mpp)#`
		Configures management plane protection to allow and disallow protocols and enters management plane protection configuration mode.

Step 4	**inband**
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpp)# inband`
	`RP/0/RP0/CPU0:router(config-mpp-inband)#`
		Configures an inband interface and enters management plane protection inband configuration mode.

	Command or Action	Purpose

Step 5	**interface** {*type instance* | **all**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpp-inband)# interface POS 0/5/0/0`
	`RP/0/RP0/CPU0:router(config-mpp-inband-POS0_5_0_0)#`
		Configures a specific inband interface or all inband interfaces as an inband interface. Use the **interface** command to enter management plane protection inband interface configuration mode.
		Among the list of interfaces, RP and SRP Ethernet interfaces cannot be configured.
		• Use the **all** keyword to configure all interfaces.

Step 6	**allow** {*protocol* | **all**} **Example:** `RP/0/RP0/CPU0:router(config-mpp-inband-POS0_5_0_0)# allow SSH`	Configures an interface as an inband interface for a specified protocol or all protocols. • Use the *protocol* argument to allow management protocols on the designated management interface. – HTTP or HTTPS – SNMP (also versions) – Secure Shell (v1 and v2) – TFTP – Telnet • Use the **all** keyword to configure the interface to allow all the management traffic that is specified in the list of protocols.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpp-inband-POS0_5_0_0)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpp-inband-POS0_5_0_0)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show mgmt-plane** [**interface** {*type instance*}] **Example:** `RP/0/RP0/CPU0:router# show mgmt-plane interface POS 0/5/0/0`	Displays information about the management plane, such as type of interface and protocols enabled on the interface. • (Optional) Use the **interface** keyword to display the details for a specific interface.

Configuration Examples for Implementing Management Plane Protection

This section provides the following configuration example:

- Configuring Management Plane Protection: Example, page SC-232

Configuring Management Plane Protection: Example

The following example shows how to configure inband interfaces under MPP:

```
configure
 control-plane
  management-plane
   inband
    interface POS0/5/0/0
     allow SSH
    !
    interface POS0/5/0/1
```

```
     allow all
     allow Telnet
    !
   !
  !
 !
show mgmt-plane

Management Plane Protection - inband interfaces
        Interface - POS0_5_0_0
                SSH Allowed

        Interface - POS0_5_0_1
                TELNET Allowed
                ALL Protocols Allowed
```

Additional References

The following sections provide references related to implementing keychain management.

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 Cisco IOS XR Software* module in *Cisco IOS XR System Security Command Reference*, Release 3.5

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

INDEX

HC	Cisco IOS XR Interface and Hardware Component Configuration Guide
IC	Cisco IOS XR IP Addresses and Services Configuration Guide
MCC	Cisco IOS XR Multicast Configuration Guide
MNC	Cisco IOS XR System Monitoring Configuration Guide
MPC	Cisco IOS XR MPLS Configuration Guide
QC	Cisco IOS XR Modular Quality of Service Configuration Guide
RC	Cisco IOS XR Routing Configuration Guide
SBC	Cisco IOS XR Session Border Controller Configuration Guide
SC	Cisco IOS XR System Security Configuration Guide
SMC	Cisco IOS XR System Management Configuration Guide

A

AAA (authentication, authorization, and accounting)
- accounting method lists, configuring **SC-210**
- accounting services, enabling **SC-217**
- authentication **SC-175**
- authentication method lists, configuring **SC-205**
- authorization, enabling **SC-216**
- authorization method lists, configuring **SC-207**
- configuration **SC-174**
- database **SC-173**
- interim accounting records, procedure **SC-214**
- login parameters, configuring **SC-220**
- per VRF (VPN routing and forwarding) **SC-196**
- RADIUS server groups, configuring **SC-201**
- remote configuration **SC-174**
- router to RADIUS server communication, configuring **SC-190**
- services, configuration (examples) **SC-221**
- TACACS+ server, configuring **SC-198**
- TACACS+ server groups, configuring **SC-203**
- task-based authorization
  - task groups **SC-172**
  - task IDs **SC-178**
  - task groups
    - configuration **SC-184**
  - user and group attributes **SC-170**
  - user groups
    - configuring **SC-186**
    - definition **SC-171**
    - inheritance **SC-171**
    - predefined **SC-171**
    - prerequisites **SC-169**
    - privilege level mapping **SC-181**
    - restrictions **SC-169**
  - users, configuring **SC-188**
  - XML schema **SC-181**

aaa accounting command **SC-214**

aaa accounting update command **SC-214**

accept-lifetime command **SC-83**

accept-tolerance command **SC-78**

accounting records, interim
- aaa accounting command **SC-214**
- aaa accounting update command **SC-214**
- procedure **SC-214**

algorithms
- See IKE, algorithms

allow command **SC-230**

antireplay window (IPSec)
- checking crypto profile **SC-117**
- crypto ipsec profile command **SC-117**
- crypto ipsec security-association replay disable command **SC-116**
- crypto ipsec security-association replay window-size command **SC-116**
- expanding and disabling globally procedure **SC-116**
- overview **SC-98**

auto-update
definition SC-29
how to configure SC-40
auto-upgrade
auto-update client command SC-40
crypto isakmp client configuration group SC-40

B

banner
banner command SC-40
crypto isakmp client configuration group command SC-40
definition SC-29
how to configure SC-40
browser-proxy
browser-proxy command SC-42
crypto isakmp client configuration browser-proxy command SC-41
crypto isakmp client configuration group command SC-42
definition SC-29
how to configure SC-41
how to map to a group SC-42
proxy command SC-41

C

CAC (call admission control)
IKE SA configuration SC-50
IKE session SC-30
overview SC-30
security association (SA) SC-30
CAs (certification authorities)
authenticating SC-10
declaring SC-8
description SC-3, SC-160
domain names, configuring (example) SC-6
host names SC-6
manual enrollment, how to cut-and-paste SC-12
RSA (Rivest, Shamir, and Adelman) key pairs
generating SC-7
supported standards SC-2
trusted point, configuring SC-8
See also certificates; CRLs; IPSec; RAs
certificates SC-3
requests SC-11
See also CAs; CRLs; RSA keys
clear crypto session command SC-32
clock set command SC-225
config-isakmp command mode, how to enable SC-35
configuration url command SC-43
configuration version command SC-43
control plane protection, MPP
control-plane command SC-230
CoPP (Control Plane Policing) SC-228
definition SC-228
cryptographic-algorithm command SC-86
crypto ipsec df-bit command SC-114
crypto ipsec pmtu command SC-139
crypto ipsec pre-fragmentation command SC-124
crypto ipsec profile command SC-117
crypto ipsec security-association idle-time command SC-121
crypto ipsec security-association replay disable command SC-116
crypto ipsec security-association replay window-size command SC-116
crypto ipsec server send-update command SC-29
crypto ipsec transform-set command SC-108
crypto isakmp client configuration browser-proxy command SC-41
crypto isakmp client configuration group command SC-40, SC-42, SC-43
crypto keyrings
configuration SC-54
guidelines and restrictions SC-54

crypto mib ipsec flowmib history failure size command SC-128
crypto nat-transparency command SC-119

D

dead-server detection
RADIUS SC-194
radius-server dead-criteria time command SC-195
radius-server dead-criteria tries command SC-195
radius-server deadtime command SC-194
deadtime command SC-202
DES (Data Encryption Standard)
definition SC-21
IKE policy parameter SC-23
description (ISAKMP peer) command SC-31
device configuration, MPP SC-230
DF (Don't Fragment) bit override
crypto ipsec df-bit command SC-114
overview SC-98
procedure SC-114
domain names
certification authority interoperability, configuring SC-6
DPD (Dead Peer Detection)
message configuration SC-63
periodic message SC-32

E

Easy VPN (Virtual Private Network) features
auto-update SC-29
banner SC-29
browser-proxy SC-29
URL configuration SC-29
encrypted nonces
See RSA encrypted nonces
encryption algorithm
See IKE, algorithms
end-time, key chain SC-76

H

hash algorithm
See IKE, algorithms
high availability overview SC-103
hitless key rollover
accept-tolerance command SC-78
procedure SC-78
host names
certification authority interoperability, configuring (examples) SC-6

I

IKE (Internet Key Exchange) security protocol
algorithms
encryption SC-35
hash SC-35
options SC-24
authentication
methods SC-24, SC-35
DH (Diffie-Hellman)
group identifier, specifying SC-35
IKE policy parameter SC-23
enabling and disabling SC-33
extended authentication SC-30
group identifier, specifying SC-35
ISAKMP identity, configuring SC-26
keys
See keys, preshared; keys, preshared using AAA server; RSA keys
mode configuration SC-28
negotiations SC-24

policies
configuring (example) SC-69
identifying SC-35
multiple SC-25
parameters SC-23, SC-24
purpose SC-23
viewing SC-36
policies, configuring SC-34
requirements
RSA encrypted nonces method SC-25
RSA signatures method SC-25
supported standards SC-21
See also IPSec; RSA encrypted nonces; SAs
IKE peer, configuration
description (ISAKMP peer) command SC-31
how to add SC-57
inband management interface, MPP
allow command SC-230
definition SC-228
inband command SC-230
interface command SC-230
interface service-gre command SC-103
interface service-ipsec command SC-103
IPSec (IPSec Network Security Protocol)
CAs
implementing with SC-5
implementing without SC-5
IPSec (IP Security)
checkpointing, description SC-98
crypto access lists SC-95
cautions, creating SC-129
creating SC-106
purpose SC-96
crypto profiles SC-94
applying to transport SC-131
applying to tunnel-ipsec interfaces SC-130
static or dynamic, configuring SC-109
dynamic crypto profiles SC-95
group policy definition
mode configuration SC-36
lifetimes
global, setting SC-105
prerequisites, implementing SC-92
transform sets
defining SC-108
description SC-96
IPSec-protected GRE virtual interface, procedure SC-136
IPSec VPN SPA
antireplay window SC-98
DF bit override overview SC-98
DPD message SC-32
how to display hardware SC-101
load balancing and high availability SC-103
NAT transparency SC-99
overview SC-101
prefragmentation SC-99
restrictions SC-93
SA idle timers SC-99
SNMP SC-101
ISAKMP SC-21
See also IKE SC-19
ISAKMP profile
considerations SC-27
locally sourced and destined traffic procedure SC-58
overview SC-26
service interfaces procedure SC-64

K

key (key chain) command SC-80
key chain
configuration (example) SC-87
configuring SC-77
end-time SC-76
key chain command SC-77
key identifier, configuring SC-79
lifetime SC-76

outbound traffic, configuring SC-84
overview SC-76
start-time SC-76
text, configuring SC-81
valid key, determining SC-82
key identifier, configuring SC-79
keyring command SC-59
keyring configuration mode, enabling SC-46
keys
mask preshared SC-27
preshared
configuring (example) SC-48
IKE policy parameter SC-23
using AAA server SC-27, SC-28
key string
how to configure SC-81
key-string command SC-82
key validation, determining SC-82

L

lifetime, key chain SC-76
load balancing overview SC-103

M

MAC (message authentication code) SC-86
authentication option SC-76
cryptographic algorithm procedure SC-85
management plane
management-plane command SC-230
MPP feature SC-229
overview SC-228
match identity command SC-59
MD5 (Message Digest 5) algorithm SC-22
IKE policy parameter SC-23
MPLS (Multiprotocol Label Switching), encapsulated packets SC-104
MPP (Management Plane Protection)
benefits SC-229
control plane protection SC-228
description SC-227, SC-229
device configuration SC-230
inband management interface SC-228
management plane SC-228
show mgmt-plane command SC-230
MTU (maximum transmission unit), default path
crypto ipsec pmtu command SC-139
procedure SC-139

N

NAT transparency (IPSec)
crypto nat-transparency command SC-119
overview SC-99
procedure SC-118
nonces
See RSA encrypted nonces

O

Oakley key exchange protocol SC-21
See also IKE
outbound traffic (key chain), configuring SC-84

P

per VRF (VPN routing and forwarding) AAA
procedure SC-196
server-private command SC-197
supported VSAs SC-196
vrf command SC-197
PFS (perfect forward secrecy)
overview SC-97
set pfs command SC-97

prefragmentation
crypto ipsec pre-fragmentation command SC-124
dependencies SC-99
overview SC-99
procedure
service-gre interfaces SC-125
service-ipsec interfaces SC-124
preshared keys
See keys, preshared; keys, preshared using AAA server
proxy command SC-41

R

RADIUS
configuring
dead-server detection SC-194
UDP ports SC-191
operation SC-183
radius-server dead-criteria time command SC-195
radius-server dead-criteria tries command SC-195
radius-server deadtime command SC-194
RAs (registration authorities)
See CAs
reverse-route command
crypto profiles SC-109
RRI (reverse-route injection) SC-127
RFC 2408, ISAKMP SC-21
RFC 2409, The Internet Key Exchange SC-21
RRI (reverse-route injection)
overview SC-100
procedure SC-127
reverse-route command SC-109, SC-127
show route command SC-127
RSA (Rivest, Shamir, and Adelman)
encrypted nonces SC-22
keys
definition SC-3
deleting SC-8
signatures SC-22
RSA (Rivest, Shamir, and Adelman) encrypted nonces
IKE policy parameter SC-23
requirements SC-24, SC-25
RSA (Rivest, Shamir, and Adelman) keys
configuring, manually SC-44
generating SC-44
peer configuration SC-46
RSA (Rivest, Shamir, and Adelman) signatures
IKE configuration SC-25
IKE policy parameter SC-23
requirements SC-24

S

SA (security association) for IPSec VPN SPA
idle timers
crypto ipsec security-association idle-time command SC-121
overview SC-99
procedure, each crypto profile SC-122
procedure, globally SC-121
lifetimes SC-121
SAM (Software Authentication Manager)
description SC-225
SAs (security associations)
lifetimes
configuring SC-35
global values, configuring SC-96
how they work SC-97
IKE policy parameter SC-23
limit overview SC-30
resource limit configuration SC-52
self-identity command SC-59
send-lifetime command SC-85
server-private command SC-197, SC-201

service interfaces procedure SC-64
service-location command SC-103
set interface tunnel-ipsec command SC-59
set pfs command SC-97
set security-association idle-time command SC-109
set security-association lifetime command SC-109
set security-association replay disable command SC-109
set session-key inbound ah command SC-110
set session-key inbound esp command SC-110
set session-key outbound ah command SC-110
set session-key outbound esp command SC-110
SHA (Secure Hash Algorithm)
  definition SC-22
  IKE policy parameter SC-23
show diag command SC-101
show key chain command SC-78
show mgmt-plane command SC-230
show radius dead-criteria host command SC-195
show route command
  IPSec-protected GRE virtual interfaces SC-136
  RRI (reverse-route injection) SC-127
  virtual interfaces (IPSec) SC-133
Skeme key exchange protocol SC-21
  See also IKE
SNMP (Simple Network Management Protocol)
  history table
    crypto mib ipsec flowmib history failure size command SC-128
    function SC-101
    how to configure SC-128
  IPSec VPN SPA SC-101
SSH (Secure Shell)
  client
    configuring SC-154
    DES and 3DES support SC-151
    description SC-151
    server support SC-151
  configuring SC-152
  prerequisites, configuring SC-150
  restrictions, implementing SC-150
  server SC-151
  SFTP
    overview SC-151
  supported versions SC-149
  troubleshooting SC-155
SSL (Secure Socket Layer)
  configuring SC-161
  description SC-159
  prerequisites, implementing SC-160
start-time, key chain SC-76
static IPSec virutal interface, procedure SC-133

T

tunnel vrf command SC-104

U

URL configuration
  configuration url command SC-43
  configuration version command SC-43
  crypto isakmp client configuration group SC-43
  how to push a mode-configuration exchange SC-43
  overview SC-29

V

virtual interfaces (IPSec)
  interface service-gre command SC-103
  interface service-ipsec command SC-103
  overview SC-102
  procedure
    IPSec-protected GRE SC-136
    static SC-133
  service-location command SC-103
  show route command SC-133, SC-136

VPN monitoring
crypto session, how to clear SC-32, SC-58
enhancements SC-31
how to add IKE peer SC-57
per-IKE peer, function SC-31
show crypto session command SC-31
summary status SC-31
vrf-aware (IPSec)
overview SC-104
tunnel vrf command SC-104
vrf command SC-104
vrf command (per VRF AAA) SC-197
VSAs (vendor-specific attributes)
per VRF AAA SC-196
supported VSAs SC-196