



System Security Configuration Guide for Cisco NCS 5000 Series Routers, IOS XR Release 6.0.x

First Published: 2015-12-23

Americas Headquarters

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



CONTENTS

PREFACE

Preface	v
Obtaining Documentation and Submitting a Service Request	v

CHAPTER 1

Configuring AAA Services	1
Configuring AAA Services	1
Prerequisites for Configuring AAA Services	2
Restrictions for Configuring AAA Services	2
References for AAA Services	26

CHAPTER 2

Implementing Keychain Management	39
Implementing Keychain Management	39
Restrictions for Implementing Keychain Management	39
Configure Keychain	39
Configure Tolerance Specification to Accept Keys	41
Configure Key Identifier for Keychain	41
Configure Text for Key String	42
Determine Valid Keys	43
Configure Keys to Generate Authentication Digest for Outbound Application Traffic	44
Configure Cryptographic Algorithm	45
Lifetime of Key	46

CHAPTER 3

Implementing Secure Shell	47
Implementing Secure Shell	47
Prerequisites to Implementing Secure Shell	48
Restrictions for Implementing Secure Shell	48
Configure SSH	49

Configure SSH Client 51

Information About Implementing Secure Shell 53

- SSH Server 53
- SSH Client 53
- SFTP Feature Overview 55
- RSA Based Host Authentication 56
- RSA Based User Authentication 56
- SSHv2 Client Keyboard-Interactive Authentication 57



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 *System Security Command Reference for Cisco NCS 5000 Series Routers*.

The preface contains the following sections:

- [Obtaining Documentation and Submitting a Service Request, on page v](#)

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*, at: <http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html>.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation as an RSS feed and delivers content directly to your desktop using a reader application. The RSS feeds are a free service.



CHAPTER 1

Configuring AAA Services

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

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

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



Note Currently, only default VRF is supported. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

- [Configuring AAA Services, on page 1](#)

Configuring AAA Services

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

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

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



Note Currently, only default VRF is supported. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

Prerequisites for Configuring AAA Services

The following are the prerequisites to configure AAA services:

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

Restrictions for Configuring AAA Services

This section lists the restrictions for configuring AAA services.

Compatibility

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

Interoperability

Router administrators can use the same AAA server software and database (for example, CiscoSecure ACS) for the router and any other Cisco equipment that does not currently run the Cisco software. To support interoperability between the router and external TACACS+ servers that do not support task IDs, see the “[Task IDs for TACACS+ and RADIUS Authenticated Users, on page 35](#)” section.

Configure Task group

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

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

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

Before you begin

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



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

SUMMARY STEPS

1. **configure**
2. **taskgroup** *taskgroup-name*
3. **description** *string*
4. **task** {**read** | **write** | **execute** | **debug**} *taskid-name*
5. Repeat for each task ID to be associated with the task group named in Step 2.
6. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

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

Step 3 **description** *string*

Example:

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

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

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

Example:

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

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

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

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

Step 6 **commit**

What to do next

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

Configure User Groups

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

Before you begin



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

SUMMARY STEPS

1. **configure**
2. **usergroup** *usergroup-name*
3. **description** *string*
4. **inherit usergroup** *usergroup-name*
5. **taskgroup** *taskgroup-name*
6. Repeat Step for each task group to be associated with the user group named in Step 2.
7. **commit**

DETAILED STEPS

Step 1 **configure**

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 submenu.

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

Step 3 **description** *string*

Example:

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

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

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

Example:

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

- Explicitly defines permissions for the user group.

Step 5 `taskgroup` *taskgroup-name*

Example:

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

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

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

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

—

Step 7 `commit`

Configure Users

Perform this task to configure a user.

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

SUMMARY STEPS

1. `configure`
2. `username` *user-name*
3. Do one of the following:
 - `password` {0 | 7} *password*
 - `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. `commit`

DETAILED STEPS

Step 1 `configure`

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 submenu.

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

Step 3 Do one of the following:

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

Example:

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

or

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

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.

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

—

Step 6 `commit`**Configure Router to RADIUS Server Communication**

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

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

RADIUS security servers are identified on the basis of their hostname or IP address, hostname and specific User Datagram Protocol (UDP) port numbers, or IP address and specific UDP port numbers. The combination of the IP address and UDP port numbers creates a unique identifier, allowing different ports to be individually defined as RADIUS hosts providing a specific AAA service. In other words, this unique identifier enables

RADIUS requests to be sent to multiple UDP ports on a server at the same IP address. If two different host entries on the same RADIUS server are configured for the same service—for example, accounting—the second host entry configured acts as an automatic switchover backup to the first one. Using this example, if the first host entry fails to provide accounting services, the network access server tries the second host entry configured on the same device for accounting services. (The RADIUS host entries are tried in the order they are configured.)

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

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



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. **commit**
9. **show radius**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

- Use the **auth-port** *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for authentication.

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

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

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

Example:

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

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

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

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

Example:

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

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

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

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

Example:

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

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

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

Example:

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

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

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

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

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

—

Step 8 **commit**

Step 9 show radius

Example:

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

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

Radius Summary Example

```
radius source-interface Mgm0/rp0/cpu0/0 vrf default
radius-server timeout 10
radius-server retransmit 2
!
! OOB RADIUS
radius-server host 123.100.100.186 auth-port 1812 acct-port 1813
key cisco123
timeout 10
retransmit 2
!
radius-server host 123.100.100.187 auth-port 1812 acct-port 1813
key cisco123
timeout 10
retransmit 2
!
aaa group server radius radgrp
server 123.100.100.186 auth-port 1812 acct-port 1813
server 123.100.100.187 auth-port 1812 acct-port 1813
!
aaa authorization exec radauthen group radgrp local
aaa authentication login radlogin group radgrp local
!
line template vty
authorization exec radauthen
login authentication radlogin
timestamp disable
exec-timeout 0 0
!
pty-pool default 0 99 line-template vty
```

Configure RADIUS Dead-Server Detection

The RADIUS Dead-Server Detection feature lets you configure and determine the criteria that is used to mark a RADIUS server as dead. If no criteria is explicitly configured, the criteria is computed dynamically on the basis of the number of outstanding transactions. The RADIUS dead-server detection configuration results in the prompt detection of RADIUS servers that have stopped responding. The prompt detection of nonresponding RADIUS servers and the avoidance of swamped and dead-to-live-to-dead-again servers result in less downtime 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. **commit**
6. **show radius dead-criteria host** *ip-addr* [**auth-port** *auth-port*] [**acct-port** *acct-port*]

DETAILED STEPS

Step 1 **configure**

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 **commit**

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

Example:

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

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

Configure TACACS+ Server

This task configures a TACACS+ server.

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

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. **commit**
9. **show tacacs**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

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

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

Example:

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

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

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

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

Example:

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

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

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

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

Example:

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

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

Step 6 **tacacs source-interface** *type instance*

Example:

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

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

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

Step 7 Repeat step 2 through step 5 for each external server to be configured.

—

Step 8 **commit**

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.

Tacacs Summary Example:

```

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

```

Configure RADIUS Server Groups

This task configures RADIUS server groups.

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

Before you begin

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** {*hostname* | *ip-address*} [**auth-port** *port-number*] [**acct-port** *port-number*]
4. Repeat step 4 for every external server to be added to the server group named in step 3.
5. **deadtime** *minutes*
6. **commit**
7. **show radius server-groups** [*group-name* [**detail**]]

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

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

Example:

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

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

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

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

Step 5 **deadtime** *minutes*

Example:

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

Configures the deadtime value at the RADIUS server group level.

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

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

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

Step 6 **commit**

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

Example:

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

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

What to do next

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

Configure TACACS+ Server Groups

This task configures TACACS+ server groups.

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

Before you begin

For successful configuration, the external server should be accessible at the time of configuration. When configuring the same IP address for global and vrf configuration, server-private parameters are required.

SUMMARY STEPS

1. **configure**
2. **aaa group server tacacs+ *group-name***
3. **server {*hostname* | *ip-address*}**
4. Repeat step 3 for every external server to be added to the server group named in step 2.
5. **commit**
6. **show tacacs server-groups**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

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

Example:

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

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

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

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

Step 5 **commit**

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.

Create Series of Authentication Methods

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



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

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

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

Before you begin



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

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

SUMMARY STEPS

1. **configure**
2. **aaa authentication {login} {default | list-name} method-list**
3. **commit**
4. Repeat Step 1 through Step 3 for every authentication method list to be configured.

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

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

Step 3 **commit**

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

Create Series of Authorization Methods

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



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

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

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

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

- Accept the request as is.
- Refuse authorization.

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.

SUMMARY STEPS

1. **configure**
2. **aaa authorization** {**commands** | **eventmanager** | **exec** | **network**} {**default** | *list-name*} {**none** | **local** | **group** {**tacacs+** | **radius** | *group-name*}}
3. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

- The **commands** keyword configures authorization for all XR EXEC mode shell commands. Command authorization applies to the EXEC mode commands issued by a user. Command authorization attempts authorization for all XR EXEC mode commands.
- The **eventmanager** keyword applies an authorization method for authorizing an event manager (fault manager).
- The **exec** keyword configures authorization for an interactive (XR EXEC mode) session.
- The **network** keyword configures authorization for network services like PPP or IKE.
- The **default** keyword causes the listed authorization methods that follow this keyword to be the default list of methods for authorization.
- A *list-name* character string identifies the authorization method list. The method list itself follows the method list name. Method list types are entered in the preferred sequence. The listed method list types can be any one of the following:
 - **none**—The network access server (NAS) does not request authorization information. Authorization always succeeds. No subsequent authorization methods will be attempted. However, the task ID authorization is always required and cannot be disabled.
 - **local**—Uses local database for authorization.

- **group tacacs+**—Uses the list of all configured TACACS+ servers for authorization. The NAS exchanges authorization information with the TACACS+ security daemon. TACACS+ authorization defines specific rights for users by associating AV pairs, which are stored in a database on the TACACS+ security server, with the appropriate user.
- **group radius**—Uses the list of all configured RADIUS servers for authorization.
- **group *group-name***—Uses a named server group, a subset of TACACS+ or RADIUS servers for authorization as defined by the **aaa group server tacacs+** or **aaa group server radius** command.

Step 3 **commit**

Create Series of Accounting Methods

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

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

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

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

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

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **aaa accounting** {**commands** | **exec** | **network**} {**default** | *list-name*} {**start-stop** | **stop-only**}
 - {**none** | *method*}
3. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 Do one of the following:

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

Example:

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

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

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

- The **commands** keyword enables accounting for XR EXEC mode shell commands.
- The **exec** keyword enables accounting for an interactive (XR EXEC mode) session.
- The **network** keyword enables accounting for all network-related service requests, such as Point-to-Point Protocol (PPP).
- The **default** keyword causes the listed accounting methods that follow this keyword to be the default list of methods for accounting.
- A *list-name* character string identifies the accounting method list.
- The **start-stop** keyword sends a “start accounting” notice at the beginning of a process and a “stop accounting” notice at the end of a process. The requested user process begins regardless of whether the “start accounting” notice was received by the accounting server.
- The **stop-only** keyword sends a “stop accounting” notice at the end of the requested user process.
- The **none** keyword states that no accounting is performed.
- The method list itself follows the **start-stop** keyword. Method list types are entered in the preferred sequence. The method argument lists the following types:
 - **group tacacs+**—Use the list of all configured TACACS+ servers for accounting.
 - **group radius**—Use the list of all configured RADIUS servers for accounting.
 - **group group-name**—Use a named server group, a subset of TACACS+ or RADIUS servers for accounting as defined by the **aaa group server tacacs+** or **aaa group server radius** command.
- The example defines a **default** command accounting method list, in which accounting services are provided by a TACACS+ security server, with a stop-only restriction.

Step 3 commit

Generate Interim Accounting Records

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



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

SUMMARY STEPS

1. **configure**
2. **aaa accounting update** {**newinfo** | **periodic** *minutes*}
3. **commit**

DETAILED STEPS**Step 1** **configure****Step 2** **aaa accounting update** {**newinfo** | **periodic** *minutes*}**Example:**

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

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

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

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

Step 3 **commit****Apply Method List**

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

SUMMARY STEPS

1. **configure**
2. **line** { **console** | **default** | **template** *template-name*}
3. **authorization** {**commands** | **exec**} {**default** | *list-name*}
4. **commit**

DETAILED STEPS**Step 1** **configure****Step 2** **line** { **console** | **default** | **template** *template-name*}**Example:**

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

Enters line template configuration mode.

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

Example:

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

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

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

Step 4 **commit**

What to do next

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

Enable Accounting Services

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

SUMMARY STEPS

1. **configure**
2. **line** { **console** | **default** | **template** *template-name*}
3. **accounting** {**commands** | **exec**} {**default** | *list-name*}
4. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

Enters line template configuration mode.

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

Example:

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

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

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

Step 4 **commit**

What to do next

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

Configure Login Parameters

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

SUMMARY STEPS

1. **configure**
2. **line template** *template-name*
3. **timeout login response** *seconds*
4. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

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

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

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

Example:

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

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

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

Step 4 `commit`

Task Maps

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

Format of the Task String

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

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



Note

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

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

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

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



Note

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

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

SUMMARY STEPS

1. Enter your username and password.
2. Click the **Group Setup** button to display the **Group Setup** window.

3. From the Group drop-down list, select the group that you want to update.
4. Click the **Edit Settings** button.
5. Use the scroll arrow to locate the Shell (exec) check box.
6. Check the **Shell (exec)** check box to enable the custom attributes configuration.
7. Check the **Custom attributes** check box.
8. Enter the following task string without any blank spaces or quotation marks in the field:
9. Click the **Submit + Restart** button to restart the server.

DETAILED STEPS

-
- Step 1** Enter your username and password.
- Step 2** Click the **Group Setup** button to display the **Group Setup** window.
- Step 3** From the Group drop-down list, select the group that you want to update.
- Step 4** Click the **Edit Settings** button.
- Step 5** Use the scroll arrow to locate the Shell (exec) check box.
- Step 6** Check the **Shell (exec)** check box to enable the custom attributes configuration.
- Step 7** Check the **Custom attributes** check box.
- Step 8** Enter the following task string without any blank spaces or quotation marks in the field:

Example:

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

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

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

Example:

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

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

Example:

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

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

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

Example:

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

References for AAA Services

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

User, User Groups, and Task Groups

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

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

User Categories

Router users are classified into the following categories:

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

Root System Users

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

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

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

Root SDR Users

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

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

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

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

Secure Domain Router (SDR) Users

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

User Groups

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

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

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

Predefined User Groups

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

- **cisco-support:** This group is used by the Cisco support team.
- **netadmin:** Has the ability to control and monitor all system and network parameters.
- **operator:** A demonstration group with basic privileges.
- **root-1r:** Has the ability to control and monitor the specific secure domain router.
- **sysadmin:** Has the ability to control and monitor all system parameters but cannot configure network protocols.
- **serviceadmin:** Service administration tasks, for example, Session Border Controller (SBC).

User-Defined User Groups

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

User Group Inheritance

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

Task Groups

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

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

Predefined Task Groups

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

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

User-Defined Task Groups

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

Group Inheritance

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

Command Access in XR and Admin Modes

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

XR Task or Group	Sysadmin VM Group	Access	Example
root-lr	Root-system group	Full access to the system configuration.	<pre>RP/0/RP0/CPU0:ios#show user group Mon Nov 3 13:48:54.536 UTC root-lr, cisco-support RP/0/RP0/CPU0:ios#show user tasks inc root-lr Mon Nov 3 13:49:06.495 UTC Task: root-lr : READ WRITE EXECUTE DEBUG (reserved) RP/0/RP0/CPU0:ios#admin sysadmin-vm:0_RP0# show aaa user-group Mon Nov 3 13:48:00.790 UTC User group : root-system</pre>
Admin-r/w/x/d	Admin-r	Read only commands on Sysadmin VM	<pre>taskgroup tg-admin-write task write admin task execute admin ! usergroup ug-admin-write taskgroup tg-admin-write ! username admin-write group ug-admin-write password admin-write ! RP/0/RP0/CPU0:ios#show user group Mon Nov 3 14:09:29.676 UTC ug-admin-write RP/0/RP0/CPU0:ios#show user tasks Mon Nov 3 14:09:35.244 UTC Task: admin : READ WRITE EXECUTE RP/0/RP0/CPU0:ios#admin Mon Nov 3 14:09:40.401 UTC admin-write connected from 127.0.0.1 using console on xr-vm_node0_RP0_CPU0 sysadmin-vm:0_RP0# show aaa user-group Mon Nov 3 13:53:00.790 UTC User group : admin-r</pre>

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

Administrative Model

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

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

Administrative Access

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

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



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

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

To avoid a lockout, we recommend one or both of the following:

- Before turning on TACACS+ command authorization or XR EXEC mode authorization on the console, make sure that the user who is configuring the authorization is logged in using the appropriate user permissions in the TACACS+ profile.

- If the security policy of the site permits it, use the **none** option for command authorization or XR EXEC mode authorization so that if the TACACS+ servers are not reachable, AAA rolls over to the **none** method, which permits the user to run the command.

AAA Database

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

Local Database

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



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

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



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

Remote Database

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

Remote AAA Configuration

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

Client Configuration

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

User Groups

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

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

Task Groups

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

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

AAA Configuration

This section provides information about AAA configuration.

Method Lists

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

Rollover Mechanism

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

The following methods are available:

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

Server Grouping

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

Authentication

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

Authentication of Non-Owner Secure Domain Router User

When logging in from a non-owner secure domain router, the root system user must add the “@admin” suffix to the username. Using the “@admin” suffix sends the authentication request to the owner secure domain router for verification. The owner secure domain router uses the methods in the list-name **remote** for choosing

the authentication method. The **remote** method list is configured using the **aaa authentication login remote method1 method2...** command.

Authentication of Owner Secure Domain Router User

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

Authentication of Secure Domain Router User

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

Authentication Flow of Control

AAA performs authentication according to the following process:

1. A user requests authentication by providing a username and password (or secret).
2. AAA verifies the user's password and rejects the user if the password does not match what is in the database.
3. AAA determines the role of the user (root SDR user, or SDR user).
 - If the user has been configured as a member of an owner secure domain router user group, then AAA authenticates the user as an owner secure domain router user.
 - If the user has not been configured as a member of an owner secure domain router user group, AAA authenticates the user as a secure domain router user.

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

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

Password Types

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

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

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

Task-based Authorization

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

Task IDs

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

Task IDs are assigned to users through the following means:

Each user is associated with one or more user groups. Every user group is associated with one or more *task groups*; in turn, every task group is defined by a set of task IDs. Consequently, a user’s association with a particular user group links that user to a particular set of task IDs. A user that is associated with a task ID can execute any operation associated with that task ID.

General Usage Guidelines for Task IDs

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



Note Restconf will be supported in a future release.

Table 1: Task ID Classes

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

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

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

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

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

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

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

Task IDs for TACACS+ and RADIUS Authenticated Users

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

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

Privilege Level Mapping

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

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

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

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

XML Schema for AAA Services

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

The following schema are published by AAA:

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

Netconf and Restconf for AAA Services

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



Note Restconf will be supported in a future release.

About RADIUS

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

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

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



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

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

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

- Networks with multiple-vendor access servers, each supporting RADIUS. For example, access servers from several vendors use a single RADIUS server-based security database. In an IP-based network with multiple vendors' access servers, dial-in users are authenticated through a RADIUS server that has been customized to work with the Kerberos security system.
- Turnkey network security environments in which applications support the RADIUS protocol, such as in an access environment that uses a "smart card" access control system. In one case, RADIUS has been used with Enigma security cards to validate users and grant access to network resources.

- Networks already using RADIUS. You can add a Cisco router with RADIUS to the network. This might be the first step when you make a transition to a Terminal Access Controller Access Control System Plus (TACACS+) server.
- Networks in which a user must access only a single service. Using RADIUS, you can control user access to a single host, utility such as Telnet, or protocol such as Point-to-Point Protocol (PPP). For example, when a user logs in, RADIUS identifies this user as having authorization to run PPP using IP address 10.2.3.4 and the defined access list is started.
- Networks that require resource accounting. You can use RADIUS accounting independent of RADIUS authentication or authorization. The RADIUS accounting functions allow data to be sent at the start and end of services, indicating the amount of resources (such as time, packets, bytes, and so on) used during the session. An Internet service provider (ISP) might use a freeware-based version of RADIUS access control and accounting software to meet special security and billing needs.
- Networks that support preauthentication. Using the RADIUS server in your network, you can configure AAA preauthentication and set up the preauthentication profiles. Preauthentication enables service providers to better manage ports using their existing RADIUS solutions and to efficiently manage the use of shared resources to offer differing service-level agreements.

Network Security Situations in Which RADIUS is Unsuitable

RADIUS is not suitable in the following network security situations:

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

RADIUS Operation

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

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

1. **CHANGE PASSWORD**—A request is issued by the RADIUS server, asking the user to select a new password.

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

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



CHAPTER 2

Implementing Keychain Management

This module describes how to implement keychain management on. Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.

- [Implementing Keychain Management, on page 39](#)

Implementing Keychain Management

This module describes how to implement keychain management on. Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.

Restrictions for Implementing Keychain Management

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

Configure Keychain

This task configures a name for the keychain.

You can create or modify the name of the keychain.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Step 2 `key chain` *key-chain-name***Example:**

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
RP/0/RP0/CPU0:router(config-isis-keys)#
```

Creates a name for the keychain.

Note Configuring only the keychain name without any key identifiers is considered a nonoperation. When you exit the configuration, the router does not prompt you to commit changes until you have configured the key identifier and at least one of the mode attributes or keychain-key configuration mode attributes (for example, lifetime or key string).

Step 3 `commit`**Step 4** `show key chain` *key-chain-name***Example:**

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

(Optional) Displays the name of the keychain.

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

Example

The following example shows how to configure keychain management:

```
configure
key chain isis-keys
accept-tolerance infinite
key 8
key-string 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]
```

Configure Tolerance Specification to Accept Keys

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

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

Example:

```
RP/0//CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

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

Example:

```
RP/0//CPU0:router(config-isis-keys)# accept-tolerance infinite
```

Configures 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 **commit**

Configure Key Identifier for 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. **commit**

DETAILED STEPS**Step 1** **configure****Step 2** **key chain** *key-chain-name***Example:**

```
RP/0//CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id***Example:**

```
RP/0//CPU0:router(config-isis-keys)# key 8
```

Creates a key for the keychain. The key ID 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 **commit****Configure Text for 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. **commit**

DETAILED STEPS**Step 1** **configure****Step 2** **key chain** *key-chain-name***Example:**

```
RP/0//CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

Step 3 **key** *key-id***Example:**


```
RP/0//CPU0:router(config-isis-keys)# key 8
RP/0//CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

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

Example:

```
RP/0//CPU0:router(config-isis-keys-0x8)# key-string password 8
```

Specifies the text string for the key.

- Use the **clear** keyword to specify the key string in clear text form; use the **password** keyword to specify the key in encrypted form.

Step 5 **commit**

Determine 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** *duration-value* | **infinite** | *end-time*]
5. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

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

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/RP0/CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

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

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/RP00/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 **commit**

Configure Keys to Generate Authentication Digest for 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** *duration-value* | **infinite** | *end-time*]
5. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
```

Creates a name for the keychain.

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

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)# key 8
RP/0/RP0/CPU0:router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

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

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys)#key 8
RP/0/RP00/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 **commit**

Configure 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. **commit**

DETAILED STEPS

Step 1 **configure**

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

Example:

```
RP/0/RP0/CPU0:router(config)# key chain isis-keys
RP/0/RP0/CPU0:router(config-isis-keys) #
```

Creates a name for the keychain.

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

Example:

```
RP/0/RP0/CPU0:router(config-isis-keys) # key 8
RP/0/RP0/CPU0:router(config-isis-keys-0x8) #
```

Creates a key for the keychain.

Step 4 **cryptographic-algorithm** [HMAC-MD5 | HMAC-SHA1-12 | HMAC-SHA1-20 | MD5 | SHA-1]

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

The routing protocols each support a different set of cryptographic algorithms:

- Border Gateway Protocol (BGP) supports only HMAC-MD5 and HMAC-SHA1-12.
- Intermediate System-to-Intermediate System (IS-IS) supports only HMAC-MD5.
- Open Shortest Path First (OSPF) supports only MD5 and HMAC-MD5.

Step 5 `commit`

Lifetime of Key

If you are using keys as the security method, you must specify the lifetime for the keys and change the keys on a regular basis when they expire. To maintain stability, each party must be able to store and use more than one key for an application at the same time. A keychain is a sequence of keys that are collectively managed for authenticating the same peer, peer group, or both.

Keychain management groups a sequence of keys together under a keychain and associates each key in the keychain with a lifetime.



Note Any key that is configured without a lifetime is considered invalid; therefore, the key is rejected during configuration.

The lifetime of a key is defined by the following options:

- Start-time—Specifies the absolute time.
- End-time—Specifies the absolute time that is relative to the start-time or infinite time.

Each key definition within the keychain must specify a time interval for which that key is activated; for example, lifetime. Then, during a given key's lifetime, routing update packets are sent with this activated key. Keys cannot be used during time periods for which they are not activated. Therefore, we recommend that for a given keychain, key activation times overlap to avoid any period of time for which no key is activated. If a time period occurs during which no key is activated, neighbor authentication cannot occur; therefore, routing updates can fail.

Multiple keychains can be specified.



CHAPTER 3

Implementing Secure Shell

Secure Shell (SSH) is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley **rexec** and **rsh** tools.

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

This module describes how to implement Secure Shell.

Feature History for Implementing Secure Shell

Release	Modification
Release 6.0	This feature was introduced.



Note Currently, only default VRF is supported. Multicast, L3VPN, VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

- [Implementing Secure Shell, on page 47](#)

Implementing Secure Shell

Secure Shell (SSH) is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley **rexec** and **rsh** tools.

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

This module describes how to implement Secure Shell.

Feature History for Implementing Secure Shell

Release	Modification
Release 6.0	This feature was introduced.



Note Currently, only default VRF is supported. Multicast, L3VPN, VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

Prerequisites to Implementing Secure Shell

The following prerequisites are required to implement Secure Shell:

- Download the required image on your router. The SSH server and SSH client require you to have a crypto package (data encryption standard [DES], 3DES and AES) from Cisco downloaded on your router.
- Configure user authentication for local or remote access. You can configure authentication with or without authentication, authorization, and accounting (AAA).
- AAA authentication and authorization must be configured correctly for Secure Shell File Transfer Protocol (SFTP) to work.

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 operates only in client mode.
- In order for SFTP to work properly, the remote SSH server must enable the SFTP server functionality. For example, the SSHv2 server is configured to handle the SFTP subsystem with a line such as `/etc/ssh2/sshd2_config`:
- **subsystem-sftp /usr/local/sbin/sftp-server**
- The SFTP server is usually included as part of SSH packages from public domain and is turned on by default configuration.
- SFTP is compatible with sftp server version OpenSSH_2.9.9p2 or higher.
- RSA-based user authentication is supported in the SSH and SFTP servers. The support however, is not extended to the SSH client.
- Execution shell and SFTP are the only applications supported.
- The SFTP client does not support remote filenames containing wildcards (*, ?, []). The user must issue the **sftp** command multiple times or list all of the source files from the remote host to download them on to the router. For uploading, the router SFTP client can support multiple files specified using a wildcard provided that the issues mentioned in the first through third bullets in this section are resolved.

- The cipher preference for the SSH server follows the order AES128, AES192, AES256, and, finally, 3DES. The server rejects any requests by the client for an unsupported cipher, and the SSH session does not proceed.
- Use of a terminal type other than vt100 is unsupported, and the software generates a warning message in this case.
- Password messages of “none” are unsupported on the SSH client.
- Because the router infrastructure does not provide support for UNIX-like file permissions, files created on the local device lose the original permission information. For files created on the remote file system, the file permission adheres to the umask on the destination host and the modification and last access times are the time of the copy.

Configure SSH

Perform this task to configure SSH.



Note Currently, only default VRF is supported.



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

SUMMARY STEPS

1. **configure**
2. **hostname** *hostname*
3. **domain name** *domain-name*
4. **commit**
5. **crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*]
6. **crypto key generate dsa**
7. **configure**
8. **ssh timeout** *seconds*
9. Do one of the following:
 - **ssh server** [**vrf** *vrf-name*]
 - **ssh server v2**
10. **commit**
11. **show ssh**
12. **show ssh session details**

DETAILED STEPS

-
- Step 1** **configure**
- 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 **commit**

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 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.

Step 9 Do one of the following:

- `ssh server [vrf vrf-name]`
- `ssh server v2`

Example:

```
RP/0/RP0/CPU0:router(config)# ssh server v2
```

- (Optional) Brings up an SSH server using a specified VRF of up to 32 characters. If no VRF is specified, the default VRF is used.

To stop the SSH server from receiving any further connections for the specified VRF, use the **no** form of this command. If no VRF is specified, the default is assumed.

Note The SSH server can be configured for multiple VRF usage.

- (Optional) Forces the SSH server to accept only SSHv2 clients if you configure the SSHv2 option by using the **ssh server v2** command. If you choose the **ssh server v2** command, only the SSH v2 client connections are accepted.

Step 10 `commit`

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.

Configure SSH Client

Perform this task to configure an SSH client.



Note Currently, only default VRF is supported.

SUMMARY STEPS

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

3. commit

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

DETAILED STEPS**Step 1** configure

Step 2 ssh client knownhost *device* : */filename*

Example:

```
RP/0/RP0/CPU0:router(config)# ssh client knownhost slot1:/server_pubkey
```

(Optional) Enables the feature to authenticate and check the server public key (pubkey) at the client end.

Note The complete path of the filename is required. The colon (:) and slash mark (/) are also required.

Step 3 commit

Step 4 ssh {*ipv4-address* | *ipv6-address* | *hostname*} [**username** *user-* **cipher** | **source-interface** *type instance*]

Enables an outbound SSH connection.

- To run an SSHv2 server, you must have a VRF. This may be the default or a specific VRF. VRF changes are applicable only to the SSH v2 server.
- The SSH client tries to make an SSHv2 connection to the remote peer. If the remote peer supports only the SSHv1 server, the peer internally spawns an SSHv1 connection to the remote server.
- The **cipher des** option can be used only with an SSHv1 client.
- The SSHv1 client supports only the 3DES encryption algorithm option, which is still available by default for those SSH clients only.
- If the *hostname* argument is used and the host has both IPv4 and IPv6 addresses, the IPv6 address is used.

- If you are using SSHv1 and your SSH connection is being rejected, 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** command.

- No domain specified

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

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



Note If you are using Putty version 0.63 or higher to connect to the SSH client, set the 'Chokes on PuTTYs SSH2 winadj request' option under SSH > Bugs in your Putty configuration to 'On.' This helps avoid a possible breakdown of the session whenever some long output is sent from IOS XR to the Putty client.

Configuring Secure Shell: 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 rsa/dsa
configure
ssh server
end
```

Information About Implementing Secure Shell

To implement SSH, you should understand the following concepts:

SSH Server

The SSH server feature enables an SSH client to make a secure, encrypted connection to a Cisco router. This connection provides functionality that is similar to that of an inbound Telnet connection. Before SSH, security was limited to Telnet security. SSH allows a strong encryption to be used with the Cisco software authentication. The SSH server in Cisco software works with publicly and commercially available SSH clients.

SSH Client

The SSH client feature is an application running over the SSH protocol to provide device authentication and encryption. The SSH client enables a Cisco router to make a secure, encrypted connection to another Cisco router or to any other device running the SSH server. This connection provides functionality that is similar to that of an outbound Telnet connection except that the connection is encrypted. With authentication and encryption, the SSH client allows for a secure communication over an insecure network.

The SSH client works with publicly and commercially available SSH servers. The SSH client supports the ciphers of AES, 3DES, message digest algorithm 5 (MD5), SHA1, and password authentication. User authentication is performed in the Telnet session to the router. The user authentication mechanisms supported for SSH are RADIUS, TACACS+, and the use of locally stored usernames and passwords.

The SSH client supports setting DSCP value in the outgoing packets.

```
ssh client dscp <value from 0 - 63>
```

If not configured, the default DSCP value set in packets is 16 (for both client and server).

The SSH client supports the following options:

- **DSCP**—DSCP value for SSH client sessions.

```
RP/0/5/CPU0:router#configure
RP/0/5/CPU0:router(config)#ssh ?
  client  Provide SSH client service
  server  Provide SSH server service
  timeout Set timeout value for SSH
RP/0/5/CPU0:router(config)#ssh client ?
```

- **Knownhost**—Enable the host pubkey check by local database.

- **Source-interface**—Source interface for SSH client sessions.

```
RP/0/5/CPU0:router(config)#ssh client source-interface ?
  ATM          ATM Network Interface(s)
  BVI          Bridge-Group Virtual Interface
  Bundle-Ether Aggregated Ethernet interface(s)
  Bundle-POS   Aggregated POS interface(s)
  CEM          Circuit Emulation interface(s)
  GigabitEthernet GigabitEthernet/IEEE 802.3 interface(s)
  IMA          ATM Network Interface(s)
  IMtestmain   IM Test Interface
  Loopback     Loopback interface(s)
  MgmtEth      Ethernet/IEEE 802.3 interface(s)
  Multilink    Multilink network interface(s)
  Null         Null interface
  PFItestmain  PFI Test Interface
  PFItestnothw PFI Test Not-HW Interface
  POS          Packet over SONET/SDH network interface(s)
  PW-Ether     PWHE Ethernet Interface
  PW-IW        PWHE VC11 IP Interworking Interface
  Serial       Serial network interface(s)
  VASILeft     VASI Left interface(s)
  VASIRight    VASI Right interface(s)
  test-bundle-channel Aggregated Test Bundle interface(s)
  tunnel-ipsec IPsec Tunnel interface(s)
  tunnel-mte   MPLS Traffic Engineering P2MP Tunnel interface(s)
  tunnel-te    MPLS Traffic Engineering Tunnel interface(s)
  tunnel-tp    MPLS Transport Protocol Tunnel interface
RP/0/5/CPU0:router(config)#ssh client source-interface
RP/0/5/CPU0:router(config)#
```

SSH also supports remote command execution as follows:

```
RP/0/5/CPU0:router#ssh ?
  A.B.C.D IPv4 (A.B.C.D) address
  WORD   Hostname of the remote node
  X:X::X IPv6 (A:B:C:D...:D) address
  vrf    vrf table for the route lookup
RP/0/5/CPU0:router#ssh 1.1.1.1 ?
  cipher      Accept cipher type
  command     Specify remote command (non-interactive)
```

```

source-interface Specify source interface
username Accept userid for authentication
<cr>
RP/0/5/CPU0:router#ssh 12.28.46.6 username admin command "show redundancy sum"
Password:

Wed Jan  9 07:05:27.997 PST
Active Node Standby Node
-----
0/4/CPU0 0/5/CPU0 (Node Ready, NSR: Not Configured)

RP/0/5/CPU0:router#

```

SFTP Feature Overview

SSH includes support for standard file transfer protocol (SFTP), a new standard file transfer protocol introduced in SSHv2. This feature provides a secure and authenticated method for copying router configuration or router image files.

The SFTP client functionality is provided as part of the SSH component and is always enabled on the router. Therefore, a user with the appropriate level can copy files to and from the router. Like the **copy** command, the **sftp** command can be used only in XR EXEC mode.

The SFTP client is VRF-aware, and you may configure the secure FTP client to use the VRF associated with a particular source interface during connections attempts. The SFTP client also supports interactive mode, where the user can log on to the server to perform specific tasks via the Unix server.

The SFTP Server is a sub-system of the SSH server. In other words, when an SSH server receives an SFTP server request, the SFTP API creates the SFTP server as a child process to the SSH server. A new SFTP server instance is created with each new request.

The SFTP requests for a new SFTP server in the following steps:

- The user runs the **sftp** command with the required arguments
- The SFTP API internally creates a child session that interacts with the SSH server
- The SSH server creates the SFTP server child process
- The SFTP server and client interact with each other in an encrypted format
- The SFTP transfer is subject to LPTS policer "SSH-Known". Low policer values will affect SFTP transfer speeds



Note In IOS-XR SW release 4.3.1 onwards the default policer value for SSH-Known has been reset from 2500pps to 300pps. Slower transfers are expected due to this change. You can adjust the lpts policer value for this punt cause to higher values that will allow faster transfers

When the SSH server establishes a new connection with the SSH client, the server daemon creates a new SSH server child process. The child server process builds a secure communications channel between the SSH client and server via key exchange and user authentication processes. If the SSH server receives a request for the sub-system to be an SFTP server, the SSH server daemon creates the SFTP server child process. For each incoming SFTP server subsystem request, a new SSH server child and a SFTP server instance is created. The SFTP server authenticates the user session and initiates a connection. It sets the environment for the client and the default directory for the user.

Once the initialization occurs, the SFTP server waits for the SSH_FXP_INIT message from the client, which is essential to start the file communication session. This message may then be followed by any message based on the client request. Here, the protocol adopts a 'request-response' model, where the client sends a request to the server; the server processes this request and sends a response.

The SFTP server displays the following responses:

- Status Response
- Handle Response
- Data Response
- Name Response



Note The server must be running in order to accept incoming SFTP connections.

RSA Based Host Authentication

Verifying the authenticity of a server is the first step to a secure SSH connection. This process is called the host authentication, and is conducted to ensure that a client connects to a valid server.

The host authentication is performed using the public key of a server. The server, during the key-exchange phase, provides its public key to the client. The client checks its database for known hosts of this server and the corresponding public-key. If the client fails to find the server's IP address, it displays a warning message to the user, offering an option to either save the public key or discard it. If the server's IP address is found, but the public-key does not match, the client closes the connection. If the public key is valid, the server is verified and a secure SSH connection is established.

The IOS XR SSH server and client had support for DSA based host authentication. But for compatibility with other products, like IOS, RSA based host authentication support is also added.

RSA Based User Authentication

One of the method for authenticating the user in SSH protocol is RSA public-key based user authentication. The possession of a private key serves as the authentication of the user. This method works by sending a signature created with a private key of the user. Each user has a RSA keypair on the client machine. The private key of the RSA keypair remains on the client machine.

The user generates an RSA public-private key pair on a unix client using a standard key generation mechanism such as ssh-keygen. The max length of the keys supported is 2048 bits, and the minimum length is 512 bits. The following example displays a typical key generation activity:

```
bash-2.05b$ ssh-keygen -b 1024 -t rsa
Generating RSA private key, 1024 bit long modulus
```

The public key must be in base64 encoded (binary) format for it to be imported correctly into the box. You can use third party tools available on the Internet to convert the key to the binary format.

Once the public key is imported to the router, the SSH client can choose to use the public key authentication method by specifying the request using the “-o” option in the SSH client. For example:

```
client$ ssh -o PreferredAuthentications=publickey 1.2.3.4
```

If a public key is not imported to a router using the RSA method, the SSH server initiates the password based authentication. If a public key is imported, the server proposes the use of both the methods. The SSH client then chooses to use either method to establish the connection. The system allows only 10 outgoing SSH client connections.

Currently, only SSH version 2 and SFTP server support the RSA based authentication.



Note The preferred method of authentication would be as stated in the SSH RFC. The RSA based authentication support is only for local authentication, and not for TACACS/RADIUS servers.

Authentication, Authorization, and Accounting (AAA) is a suite of network security services that provide the primary framework through which access control can be set up on your Cisco router or access server.

SSHv2 Client Keyboard-Interactive Authentication

An authentication method in which the authentication information is entered using a keyboard is known as keyboard-interactive authentication. This method is an interactive authentication method in the SSH protocol. This type of authentication allows the SSH client to support different methods of authentication without having to be aware of their underlying mechanisms.

Currently, the SSHv2 client supports the keyboard-interactive authentication. This type of authentication works only for interactive applications.



Note The password authentication is the default authentication method. The keyboard-interactive authentication method is selected if the server is configured to support only the keyboard-interactive authentication.
