Configuring AAA Services

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

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

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

- Prerequisites for Configuring AAA Services, on page 1
- Restrictions for Configuring AAA Services, on page 1
- Information About Configuring AAA Services, on page 2
- How to Configure AAA Services, on page 20

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 15 section.

Information About Configuring AAA Services

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

User, User Groups, and Task Groups

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

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

User Categories

Router users are classified into the following categories:

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

Root System Users

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

User Groups

User groups that are created in an external server are not related to the user group concept that is used in the context of local AAA database configuration on the router. The management of external TACACS+ server or RADIUS server user groups is independent, and the router does not recognize the user group structure.

The remote user or group profiles may contain attributes that specify the groups (defined on the router) to which a user or users belong, as well as individual task IDs. For more information, see the Task IDs for TACACS+ and RADIUS Authenticated Users, on page 15section.

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

Predefined User Groups

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

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

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

User-Defined User Groups

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

User Group Inheritance

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

Task Groups

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

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

Predefined Task Groups

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

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

User-Defined Task Groups

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

Group Inheritance

Task groups support inheritance from other task groups. (Similarly, a user group can derive attributes from another user group. See the User Groups, on page 2 section.) For example, when task group A inherits task group B, the new set of attributes of task group A is the union of A and B.

Administrative Model

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

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

Administrative Access

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

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

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

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

To avoid a lockout, we recommend these:

- Before turning on TACACS+ command authorization or XR EXEC mode authorization on the console, make sure that the user who is configuring the authorization is logged in using the appropriate user permissions in the TACACS+ profile.
- If the security policy of the site permits it, use the **none** option for command authorization or XR EXEC mode authorization so that if the TACACS+ servers are not reachable, AAA rolls over to the **none** method, which permits the user to run the command.
- Make sure to allow local fallback when configuring AAA. See, Authorization Configuration, on page 66.
- If you prefer to commit the configuration on a trial basis for a specified time, you may do so by using the **commit confirmed** command, instead of direct **commit**.

AAA Database

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

Local Database

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

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

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

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

Remote Database

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

Remote AAA Configuration

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

Client Configuration

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

User Groups

User groups that are created in an external server are not related to the user group concept that is used in the context of local AAA database configuration on the router. The management of external TACACS+ server or RADIUS server user groups is independent, and the router does not recognize the user group structure. The remote user or group profiles may contain attributes that specify the groups (defined on the router) to which a user or users belong, as well as individual task IDs. For more information, see the Task IDs for TACACS+ and RADIUS Authenticated Users, on page 15section.

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

Task Groups

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

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

AAA Configuration

This section provides information about AAA configuration.

Method Lists

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

Rollover Mechanism

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

The following methods are available:

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

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

Server Grouping

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

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

Authentication

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

Authentication of Root System User

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

Authentication Flow of Control

AAA performs authentication according to the following process:

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

   - If the user has been configured as a member of an owner secure domain router user group, then AAA authenticates the user as an owner secure domain router user.

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

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

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

Korn Shell Authentication

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

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

Bypassing ksh Authentication

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

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

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

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

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

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

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

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

Authentication Failure

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

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

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

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

Password Types

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

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

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

Type 8 and Type 9 Encryption Methods

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

respect to each username. Thus, in scenarios where a lot of confidential data need to be maintained, these encryption methods ensure that the admin and other user passwords are strongly protected.

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

For more information about configuring users with Type 8 and Type 9 encryption methods, see Configure Users, on page 26 section.

Type 10 Password Encryption for User Management

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

Restrictions for Type 10 Password Encryption Usage

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

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

AAA Password Security for FIPS Compliance

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

High Availability for AAA Password Security Policy

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

To configure AAA password policy, see Configure AAA Password Policy, on page 31.

AAA Password Security Policies

AAA password security for FIPS compliance consists of these policies:

Password Composition Policy

Passwords can be composed by any combination of upper and lower case alphabets, numbers and special characters that include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")". Security administrator can also set the types and number of required characters that comprise the password, thereby providing more flexibility

for password composition rules. The minimum number of character change required between passwords is 4, by default. There is no restriction on the upper limit of the number of uppercase, lowercase, numeric and special characters.

Password Length Policy

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

Password Lifetime Policy

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

Password Expiry Policy

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

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

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

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

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

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

Password Change Policy

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

- When the security administrator needs to change the password
- When the user is trying to get authenticated using a profile and the password for the profile is expired

- When the security administrator modifies the password policy which is associated to the user, and does not immediately change the password according to the policy

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

When the security administrator changes the password security policy, and if the existing profile does not meet the password security policy rules, no action is taken if the user has already logged in to the system. In this scenario, the user is prompted to change the password when he tries to get authenticated using the profile which does not meet the password security rules.

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

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

Service Provision after Authentication

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

User Re-authentication Policy

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

User Authentication Lockout Policy

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

This is a sample syslog when user is locked out:

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

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

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

Password Policy Creation, Modification and Deletion

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

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

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

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

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

Minimum Password Length for First User Creation

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

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

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

For more information about how to configure a first user, see Configure First User on Cisco Routers, on page 24.

Password Policy for User Secret

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

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

To configure password policy for user secret, see Configure Password Policy for User Secret and Password, on page 33.

Task-based Authorization

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

Task IDs

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

Task IDs are assigned to users through the following means:

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

General Usage Guidelines for Task IDs

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

Table 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 submode. For example, to execute the **show redundancy** command, a user needs to be associated to the system (read) task ID and operations as shown in the following example:

```
Router# show redundancy
```

Task IDs for TACACS+ and RADIUS Authenticated Users

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

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

Task Maps

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

Format of the Task String

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

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

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

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

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

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

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

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

Procedure

Step 1 Enter your username and password.

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

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

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

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

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

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

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

Example:

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

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

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

Example:

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

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

Example:

```
Username:user1
Password:
Router# show user tasks

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

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

Example:

```
Username:user2
Password:
Router# show user tasks
No task ids available
```

Privilege Level Mapping

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

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

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

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

XML Schema for AAA Services

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

The following schema are published by AAA:

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

Netconf and Restconf for AAA Services

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

Note Restconf will be supported in a future release.

About RADIUS

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

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

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

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

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

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

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

Network Security Situations in Which RADIUS is Unsuitable

RADIUS is not suitable in the following network security situations:

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

RADIUS Operation

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

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

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

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

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

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

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

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

How to Configure AAA Services

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

Configure Task group

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

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

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

Before you begin

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

Step 3 **description** *string*

Example:

```
Router(config-tg)# description this is a sample task group description
```

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

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

Example:

```
Router(config-tg)# task read bgp
```

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

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

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

—

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

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

end —Prompts user to take one of these actions:

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

What to do next

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

Task Group Configuration

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

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

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

Before you begin

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

Step 3 **description** *string*

Example:

```
Router(config-tg)# description this is a sample task group description
```

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

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

Example:

```
Router(config-tg)# task read bgp
```

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

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

Step 5 Repeat Step 4 for each task ID to be associated with the task group named in Step 2.

—

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

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

end —Prompts user to take one of these actions:

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

What to do next

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

Configure User Groups

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

Before you begin

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# usergroup beta
```

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

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

Step 3 **description** *string*

Example:

```
Router(config-ug)#
description this is a sample user group description
```

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

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

Example:

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

- Explicitly defines permissions for the user group.

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

Example:

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

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

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

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

—

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

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

end —Prompts user to take one of these actions:

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

Configure First User on Cisco Routers

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

Step 1. Establish a connection to the Console port.

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

```
Enter root-system username
```

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

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

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

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

```
Enter secret
```

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

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

```
Enter secret again
```

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

Step 5. Log in to the router.

Establishes your access rights for the router management session.

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

For more information on minimum password length, see Minimum Password Length for First User Creation, on page 13.

Example

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

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

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

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

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

Configure Users

Perform this task to configure a user.

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

Note You must not use the following words as usernames:

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

- www-data

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# username user1
```

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

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

Step 3 Do one of the following:

- **password** {**0** | **7**} *password*
- **secret** {**0** | **5**|**8** | **9**| **10**} *secret*

Example:

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

or

```
Router(config-un)# secret 0 sec1
```

Specifies a password for the user named in step 2.

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

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

Example:

```
Router(config-un)# group sysadmin
```

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

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

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

—

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

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

end —Prompts user to take one of these actions:

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

Password Masking For Type 7 Password Authentication

Table 2: Feature History Table

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

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

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

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

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

Enter password:
Re-enter password:

Router(config)#commit
```

View the encrypted password:

```
Router# show run aaa
..

username us3
 password 7 105A1D0D
```

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

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

```
Enter password:
Re-enter password:


Router(config)#commit
```

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

Configure Type 8 and Type 9 Passwords

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

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

The Type 5, Type 8, Type 9 and Type 10 encryption methods provide the above mentioned options for users to configure their passwords.

For more information about configuring users with Type 8 and Type 9 encryption methods, see Configure Users, on page 26 section.

Configuration Example

Directly configuring a Type 8 encrypted password:

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

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

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

Directly configuring a Type 9 encrypted password:

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

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

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

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

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

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

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

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

Enter secret:
```

```
Re-enter secret:

Router(config)# commit
```

View the encrypted secret:

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

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

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

Enter secret:
Re-enter secret:

Router(config)# commmit
```

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

Related Topics

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

Associated Commands

- secret
- username

Configure Type 10 Password Encryption

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

Configuration Example

The Type 10 encryption is applied by default when you create a user with a clear-text password.

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

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

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

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

```
Router#configure
```

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

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

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

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

Running Configuration

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

Related Topics

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

Associated Commands

- username
- secret

Configure AAA Password Policy

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

Configuration Example

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

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

```
Router(config)#username user1 password-policy test-policy password 0 pwd1
```

Running Configuration

```
aaa password-policy test-policy
 min-length 8
 max-length 15
 lifetime months 3
 min-char-change 5
 authen-max-attempts 3
 lockout-time days 1
 !
```

Verification

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

```
Router#show aaa password-policy

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

Password Masking For AAA Password Policies

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

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

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

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

Enter password:
```

```
Re-enter password:

Router(config)#commit
```

View the encrypted password:

```
Router# show run aaa
..

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

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

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

Enter password:
Re-enter password:


Router(config)#commit
```

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

Related Topic

- AAA Password Security for FIPS Compliance, on page 10

Associated Commands

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

Configure Password Policy for User Secret and Password

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

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

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

Guidelines to Configure Password Policy for User Secret

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

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

Configuration Example

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

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

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

Running Configuration

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

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

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

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

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

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

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

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

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

  password 7 1533292F200F2D
!
```

Related Topics

- Password Policy for User Secret, on page 13

Associated Commands

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

Display Username for Failed Authentication for Telnet Protocols

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Display Username for Failed Authentication for Telnet Protocols	Release 7.10.1	With this feature, we have enhanced the security of the routers and introduced better tracking functionality to the router. The failed authentication sys log now displays the details of users who tried to log in but failed due to authentication failure. With this feature provisioned, the router can now display the user ID of both SSH and Telnet protocols. In earlier releases, this feature was available only for SSH protocols. This feature introduces the following change: **CLI**: **aaa display-login-failed-users**. **YANG DATA Model**: New XPaths for `Cisco-IOS-XR-um-aaa-task-user-cfg` (see Github, YANG Data Models Navigator)

Effective Cisco IOS XR Software Release 7.10.1, you can track the username of the users who tried to login to the router and their authentication failed in the failed authentication system logs. Prior to this release, this feature was available for SSH clients only. Now, this functionality is available for both SSH and Telnet clients. By default, the feature is disabled. When this feature is disabled, failed authentication sys logs displays the username as **unknown** for both SSH and Telnet. Once the feature is enabled, the failed authentication sys logs display the username of the users who tried to login to the router, and the login attempt was unsuccessful due to failed authentication.

Use the **aaa display-login-failed-users** command in XR Config mode to enable this feature.

Enable Display of Username for Failed Authentication

Configuration Example

```
Router#conf
Router(config)#aaa display-login-failed-users
Router(config)#commit
```

Running configuration

```
Router# show run aaa display-login-failed-users
!
aaa display-login-failed-users
!
```

Verification

This section shows example from sys logs where the user name is displayed for failed authentication after the configuration of this feature.

System logs for Telnet client:

```
RP/0/RP0/CPU0:Jul 18 14:46:31.590 UTC: exec[66608]:
%SECURITY-LOGIN-4-AUTHEN_FAILED : Failed authentication attempt by
user lab from 'console' on 'con0_RP0_CPU0'
```

System logs for SSH client:

```
RP/0/RP0/CPU0:Jul 18 14:47:51.590 UTC: ssh_syslog_proxy[1216]:
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[13519]: Failed authentication/pam
for lab from 192.168.122.1 port 44822 ssh2
```

Password Policy to Restrict Consecutive Characters

Table 4: Feature History Table

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

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

Certain key aspects of this feature are:

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

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

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

How to Restrict Consecutive Characters for User Passwords and Secrets

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

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

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

Configuration Example

Enabling the feature using CLI:

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

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

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

Applying the password policy to the user profile:

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

Running Configuration

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

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

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

Verification

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

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

```
  Restrict Old count : 0
  Restrict Username : 0
  Restrict Username Reverse  : 0
  Restrict Password Reverse  : 0
  Restrict Password Advanced : 0
  Restrict Consecutive Character :
    English Alphabet characters: 4
    English Alphabet Cyclic Wrap: True
    Qwerty Keyboard characters: 6
    Qwerty Keyboard Cyclic Wrap: False
Router#
```

Password or Secret Configuration Failure Scenarios:

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

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

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

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

Enter secret:
Re-enter secret:

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

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

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

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

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

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
```

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

To learn more about the data models and to put them to use, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Configure Router to RADIUS Server Communication

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

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

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

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

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

You can configure a maximum of 30 global RADIUS servers.

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# radius-server host host1
```

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

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

Note
The key is a text string that must match the encryption key used on the RADIUS server. Always configure the key as the last item in the **radius-server host** command syntax because the leading spaces are ignored, but spaces within and at the

end of the key are used. If you use spaces in your key, do not enclose the key in quotation marks unless the quotation marks themselves are part of the key.

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

Example:

```
Router(config)# radius-server retransmit 5
```

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

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

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

Example:

```
Router(config)# radius-server timeout 10
```

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

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

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

Example:

```
Router(config)# radius-server key 0 samplekey
```

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

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

Example:

```
Router(config)# radius source-interface 0/3/0/1
```

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

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

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

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

—

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

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

end —Prompts user to take one of these actions:

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

Step 9 show radius

Example:

```
Router# show radius
```

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

Radius Summary Example

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

Configure RADIUS Dead-Server Detection

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

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

In addition, you can configure the number of consecutive timeouts that must occur on the router before the RADIUS server is marked as dead. If the server performs both authentication and accounting, both types of packets are included in the number. Improperly constructed packets are counted as though they are timeouts.

Only retransmissions are counted, not the initial transmission. For example, each timeout causes one retransmission to be sent.

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# radius-server deadtime 5
```

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

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

Example:

```
Router(config)# radius-server dead-criteria time 5
```

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

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

Example:

```
Router(config)# radius-server dead-criteria tries 4
```

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

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

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

end —Prompts user to take one of these actions:

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

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

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

Example:

```
Router# show radius dead-criteria host 172.19.192.80
```

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

Configure Per VRF AAA

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

New Vendor-Specific Attributes (VSAs)

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

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

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

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

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

Table 5: Supported VSAs for Per VRF AAA

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

VSA Name	Value Type	Description

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

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

This task configures RADIUS server groups per VRF. For information about configuring TACACS+ server groups per VRF, refer Configure TACACS+ Server Groups, on page 57.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

Example:

IP address in IPv4 format

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

Example:

IP address in IPv6 format

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

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

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

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

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

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

Example:

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

Configures the VRF reference of an AAA RADIUS server group.

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

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

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

end —Prompts user to take one of these actions:

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

Configure TACACS+ Server

This task configures a TACACS+ server.

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

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

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

Example:

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

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

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

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

Example:

```
Router(config)# tacacs-server host 209.165.200.226 key 0 a_secret
```

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

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

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

Example:

```
Router(config)# tacacs-server host 209.165.200.226 single-connection
```

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

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

Example:

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

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

The range is:

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

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

Example:

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

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

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

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

—

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

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

end —Prompts user to take one of these actions:

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

Step 10 **show tacacs**

Example:

```
Router# show tacacs
```

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

Tacacs Summary Example:

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

Configure Authorization for a TACACS+ Server

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

Example:

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

Configure the AAA system to not perform any authorization.

Example:

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

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

Step 3 **confdConfig aaa authorization enabled**

Example:

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

Configure ConfD to perform remote authorization.

Step 4 **confdConfig aaa authorization callback enabled**

Example:

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

Configure ConfD to invoke application callbacks for authorization.

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

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

end —Prompts user to take one of these actions:

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

Configure Authentication for a TACACS+ Server

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **confdConfig aaa externalAuthentication enabled**

Example:

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

Configure Confd to perform external authentication.

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

Example:

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

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

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

Example:

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

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

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

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

end —Prompts user to take one of these actions:

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

Configure Accounting for a TACACS+ Server

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **aaa accounting command tacacs**

Example:

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

Configure remote accounting commands.

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

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

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

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

Configure RADIUS Server Groups

This task configures RADIUS server groups.

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

You can configure a maximum of:

- 30 servers per RADIUS server group
- 30 private servers per RADIUS server group

Before you begin

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

Example:

```
Router(config-sg-radius)# server 192.168.20.0
```

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

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

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

—

Step 5 **deadtime** *minutes*

Example:

```
Router(config-sg-radius)# deadtime 1
```

Configures the deadtime value at the RADIUS server group level.

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

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

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

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

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

end —Prompts user to take one of these actions:

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

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

Example:

```
Router# show radius server-groups
```

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

What to do next

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

Configure TACACS+ Server Groups

This task configures TACACS+ server groups.

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

You can configure a maximum of :

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

Before you begin

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# aaa group server tacacs+ tacgroup1
```

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

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

Example:

```
Router(config-sg-tacacs+)# server 192.168.100.0
```

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

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

Step 4 (Optional) **vrf** *vrf-id*

Example:

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

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

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

—

Step 6 (Optional) vrf *vrf-id*

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

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

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

end —Prompts user to take one of these actions:

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

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

Step 8 **show tacacs server-groups**

Example:

```
Router# show tacacs server-groups
```

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

Configure Per VRF TACACS+ Server Groups

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

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

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

Prerequisites

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

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

Configuration Example

```
Router#configure

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

Router(config)# aaa group server tacacs+ tacgroup1

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

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

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

Running Configuration

```
aaa group server tacacs+ tacgroup1
 vrf test-vrf
 server-private 10.1.1.1 port 49
  key 7 0822455D0A16
 !
 server-private 10.1.1.2 port 49
  key 7 05080F1C2243
 !
 server-private 2001:db8:1::1 port 49
  key 7 045802150C2E
 !
 server-private 2001:db8:1::2 port 49
  key 7 13061E010803
 !
!
```

Verify Per VRF TACACS+ Server Groups

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

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

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

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

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

Associated Commands

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

View TACACS+ information in Router

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
View TACACS+ information in Router	Release 7.5.4	With this feature, you can view TCP connection statistics like failures, timeout, and disconnect in connections, number of AAA packets received from an external server or sent to an external server, and so on during TACACS+ transactions. This information helps you monitor TACACS+ health in the routers. It is also helpful in identifying and debugging TACACS+ transaction failures if any. This feature introduces the following commands: • **show tacacs counters** • **show tacacs details** • **show tacacs source-interface** • **clear tacacs counters**

You can see the record of the number of requests, timeouts, failures, errors, and success for each TACACS+ server for all the AAA services using the following:

```
Router:ios# show tacacs counters

TACACS+ Server:   10.105.236.101/4010 [global]

  Authentication:
   10 requests, 4 accepts, 3 failure, 2 error, 1 timeout

  Exec Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Command Authorization:
   6 requests, 6 accepts, 0 denied, 0 error, 0 timeout

  Exec Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

  Command Accounting:
   6 requests, 6 accepts, 0 fail, 0 error, 0 timeout

TACACS+ Server:   10.105.236.101/2201 [private] vrf = default

  Authentication:
   0 requests, 0 accepts, 0 failure, 0 error, 0 timeout
```

```
  Exec Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Command Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Exec Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

  Command Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout
```

You can view the complete TACACS+ statistics including server group, source-interface, individual server statistics, inpacket, outpacket, connection open and connection close counters, and TCP connection related counters using the following:

```
Router:ios# show tacacs details

TACACS+ Server                                                  : 10.105.236.101/4010
[Global]
    Family                                                      : IPv4
    Timeout(in secs)                                            : 3
    Connection Opens                                            : 8
    Connection Closes                                           : 8
    Requests sent                                               : 6
    Response received                                           : 6
    Packets Abort                                               : 2
    Server State                                                : Down
    Server On-Hold                                              : True
    Tacacs-Single-Connect                                       : False
    Tacacs-Single-Connect-Idle-Timeout(in secs)                 : 0
    Last Connection Attempted                                   : 08:32:43 UTC Tue Aug
02 2022


TACACS+ Server                                                  : 10.105.236.101/8010
[Private] vrf=default
    Family                                                      : IPv4
    Timeout(in secs)                                            : 3
    Connection Opens                                            : 8
    Connection Closes                                           : 7
    Requests sent                                               : 7
    Response received                                           : 7
    Packets Abort                                               : 0
    Server State                                                : Up
    Server On-Hold                                              : False
    Tacacs-Single-Connect                                       : False
    Tacacs-Single-Connect-Idle-Timeout(in secs)                 : 0
    Last Connection Attempted                                   : 08:32:52 UTC Tue Aug
02 2022


TACACS+ Server-groups:

Global list of servers
    Server 10.105.236.101/4010 family=IPv4
Server group 'tac1' has 1 servers
    Servers in this group are under 'default' vrf
    Server 10.105.236.101/8010 [private] family=IPv4

TACACS+ Source-Interface:

Interface                         VRF Id                        IPV4-Address
GigabitEthernet0/0/0/0           0x60000001                        0.0.0.0
```

```
MgmtEth0/RP0/CPU0/0               0x60000000                    192.168.122.222

Interface                          VRF Id                        IPV6-Address
GigabitEthernet0/0/0/0             0x60000001                          ::
MgmtEth0/RP0/CPU0/0                0x60000000                          ::
```

You can view the TACACS+ source interface details using the following:

```
Router:ios# show tacacs source-interfaces

Interface                                          VRF Id
IPV4-Address

MgmtEth0/RP0/CPU0/0                               0x60000000
192.168.122.222

Interface                                           VRF Id
IPV6-Address

MgmtEth0/RP0/CPU0/0                               0x60000000
::
```

You can clear all AAA services counters in **show tacacs counters** command for all TACACS+ servers using the **clear tacacs counters** command:

```
Router:ios# show tacacs counters

TACACS+ Server:   10.105.236.101/4010 [global]

  Authentication:
   10 requests, 4 accepts, 3 failure, 2 error, 1 timeout

  Exec Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Command Authorization:
   6 requests, 6 accepts, 0 denied, 0 error, 0 timeout

  Exec Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

  Command Accounting:
   6 requests, 6 accepts, 0 fail, 0 error, 0 timeout

TACACS+ Server:   10.105.236.101/2201 [private] vrf = default

  Authentication:
   0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

  Exec Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Command Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Exec Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

  Command Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

Router:ios# clear tacacs counters
Router:ios# show tacacs counters

TACACS+ Server:   10.105.236.101/4010 [global]
```

```
  Authentication:
   0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

  Exec Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Command Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Exec Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

  Command Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

TACACS+ Server:   10.105.236.101/2201 [private] vrf = default

  Authentication:
   0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

  Exec Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Command Authorization:
   0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

  Exec Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

  Command Accounting:
   0 requests, 0 accepts, 0 fail, 0 error, 0 timeout
```

Configure AAA Method Lists

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

This section contains the following procedures:

Configuring Authentication Method Lists

This task configures method lists for authentication.

Authentication Configuration

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

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

The authentication can be applied to tty lines through use of the **login authentication** line configuration submode command.

Create Series of Authentication Methods

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

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

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

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

Before you begin

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# aaa authentication login default group tacacs+
```

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

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

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

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

end —Prompts user to take one of these actions:

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

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

—

Configuring Authorization Method Lists

This task configures method lists for authorization.

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

Authorization Configuration

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

selects the next method listed in the method list. This process continues until there is successful communication with a listed authorization method, or until all methods defined have been exhausted.

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

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

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

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

- **XR EXEC mode authorization—**Applies authorization for starting XR EXEC mode session.
- **Network authorization—**Applies authorization for network services, such as IKE.
- **Eventmanager authorization—**Applies an authorization method for authorizing an event manager (fault manager). RADIUS servers are not allowed to be configured for the event manager (fault manager) authorization. You are allowed to use TACACS+ or localdd.

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

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

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

- Accept the request as is.
- Refuse authorization.

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

Create Series of Authorization Methods

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

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

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

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

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

- Accept the request as is.
- Refuse authorization.

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

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

For example,

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

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

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

end —Prompts user to take one of these actions:

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

Configuring Accounting Method Lists

This task configures method lists for accounting.

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

Accounting Configuration

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

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

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

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

Create Series of Accounting Methods

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

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

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

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

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

Example:

```
Router(config)# aaa accounting commands default stop-only group tacacs+
```

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

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

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

- The example defines a **default** command accounting method list, in which accounting services are provided by a TACACS+ security server, with a stop-only restriction.

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

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

end —Prompts user to take one of these actions:

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

Generate Interim Accounting Records

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# aaa accounting update periodic 30
```

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

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

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

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

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

end —Prompts user to take one of these actions:

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

Applying Method Lists for Applications

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

This section contains the following procedures:

Enabling AAA Authorization

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

Method List Application

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

Enters line template configuration mode.

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

Example:

```
Router(config-line)# authorization commands listname5
```

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

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

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

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

end —Prompts user to take one of these actions:

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

What to do next

After applying authorization method lists by enabling AAA authorization, apply accounting method lists by enabling AAA accounting. (See the Enable Accounting Services, on page 74section.)

Enable Accounting Services

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

Enters line template configuration mode.

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

Example:

```
Router(config-line)# accounting commands listname7
```

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

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

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

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

end —Prompts user to take one of these actions:

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

What to do next

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

Configure Login Parameters

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

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

Example:

```
Router(config-line)# timeout login response 20
```

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

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

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

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

end —Prompts user to take one of these actions:

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

Command Accounting

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

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

Configuring Command Accounting

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

1. Configuring command accounting alone

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

2. Configuring command accounting along with other accounting methods

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

Command Authorization Using Local User Account

Table 7: Feature History Table

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

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

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

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

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

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

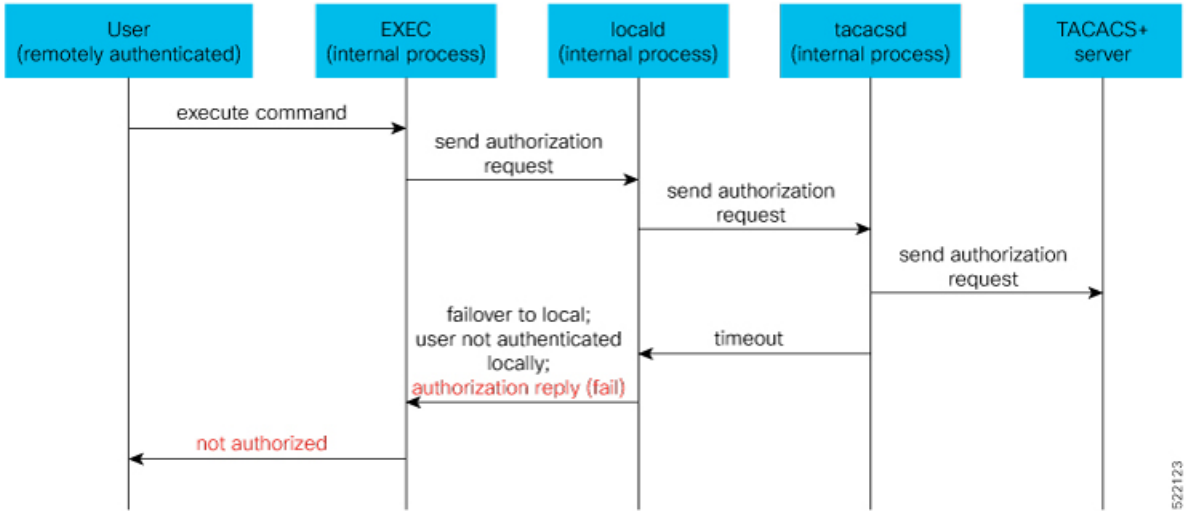
  If TACACS+ server is not reachable, then the system does not authorize the command at all. Because the user then cannot execute any command, the router gets locked out.

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

Call Flow of Command Authorization

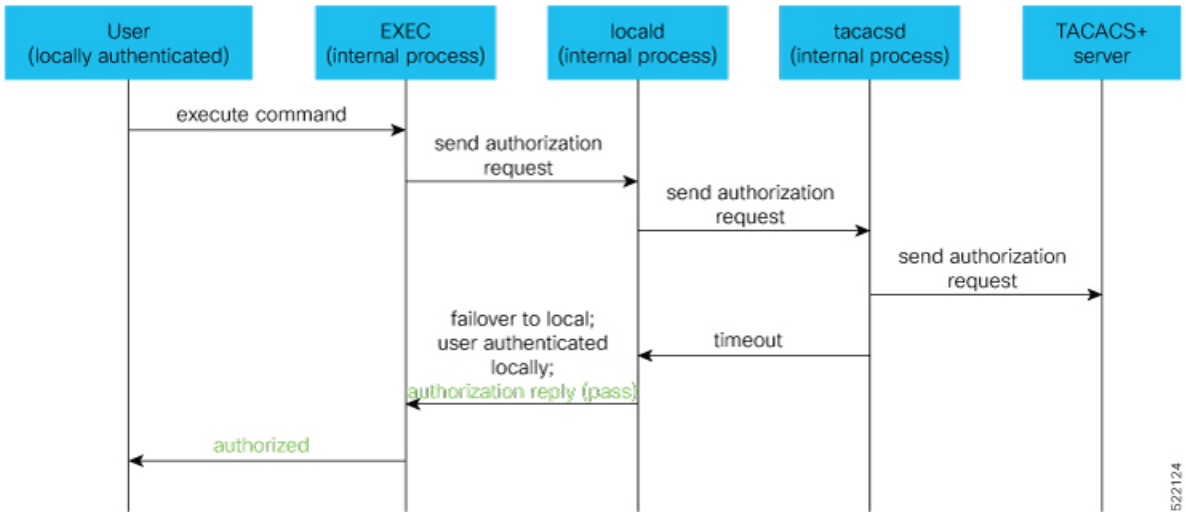
Consider a scenario where the user is remotely authenticated. In the event of timeout from the TACACS+ server, the command authorization fails. The user cannot execute any command until the TACACS+ server is reachable again, thereby preventing misuse of user privileges on the router.

Figure 1: Call Flow of Command Authorization for Remotely Authenticated Users



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

Figure 2: Call Flow of Command Authorization for Locally Authenticated Users



Configure Command Authorization Using Local User Account

Guidelines

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

Configuration Example

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

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

Or

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

Running Configuration

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

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

Verification

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

Feature Behavior and Use Case Scenarios

Feature Behavior With Various Local Command Authorization Options

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

Table 8: Feature Behavior with Various Local Command Authorization Options

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

Use Case Scenarios of Command Authorization

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

Table 9: Use Case Scenarios of Command Authorization

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

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

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

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

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

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

Configuration Example for AAA Services

The following examples show how to configure AAA services.

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

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

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

```
aaa authentication ppp default local
```

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

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

username user2
secret lab
exit
```

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

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

usergroup uga
taskgroup tga
description usergroup uga
exit
```

Username user2 is configured to inherit from user group uga.

```
username user2
group uga
exit
```

Three TACACS servers are configured.

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

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

```
usergroup priv5
taskgroup operator
exit
```

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

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

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

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

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

```
aaa default-taskgroup tga
```

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

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

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

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