Configure AAA

AAA can be implemented on NCS 1004 using administrative models of task-based authorization. Implementation involves configuring TACACS+ and RADIUS servers and groups to control user access in the software system.

Note From Release 24.4.1, the AAA local database supports configuring up to 3000 usernames. Although you can configure more than 3000 users, it may impact the system's scale and performance, which are not assured beyond this limit.

RADIUS security protocol

RADIUS is a network protocol that provides centralized authentication, authorization, and accounting management for users who connect and use a network service. The protocol operates on a client-server model.

- RADIUS clients operate on network devices such as NCS 1004.
- RADIUS clients send authentication and accounting requests to the central RADIUS server.
- RADIUS servers contain all the user authentication and network service access information.

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

Configure RADIUS server groups

Before you begin

Ensure that the external server is accessible at the time of configuration.

RADIUS server groups allow you to organize external RADIUS servers into logical groups that can be referenced from AAA method lists for authentication, authorization, or accounting services.

You can enter one or more **server** commands. The **server** command specifies the hostname or IP address of an external RADIUS server and its port numbers. After configuration, the server group is referenced from the AAA method lists for authentication, authorization, or accounting.

You can configure up to 30 servers and private servers for each RADIUS server group.

Procedure

Step 1 Configure RADIUS server parameters. For details, see Configure RADIUS server parameters.

Step 2 Enable AAA authentication for the RADIUS server group. For details, see Enable RADIUS authentication.

Configure RADIUS server parameters

This task configures the basic RADIUS server group and adds external RADIUS servers with their IP addresses, authentication ports, accounting ports, and encryption keys.

Procedure

Step 1 Enter global configuration mode.

Example:

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

Step 2 Enter the **aaa group server radius** command to create a RADIUS server group and enter server group configuration mode.

Example:

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

This command groups different server hosts into distinct lists.

Step 3 Enter the **radius-server host** command to specify the hostname or IP address of the RADIUS server host.

Example:

```
RP/0/RP0/CPU0:ios(config)# radius-server host 192.168.20.0
```

Step 4 Enter the **auth-port** command to specify the User Datagram Protocol (UDP) destination port for authentication requests.

Example:

```
RP/0/RP0/CPU0:ios(config)# auth-port 1812
```

If you set the value to 0, the host is not used for authentication. If you do not specify a value, the port number is 1645 by default.

Step 5 Enter the **acct-port** command to specify the UDP destination port for accounting requests.

Example:

```
RP/0/RP0/CPU0:ios(config)# acct-port 1813
```

If you set the value to 0, the host is not used for accounting. If you do not specify a value, the port number is 1646 by default.

Step 6 Enter the **key** command to specify the authentication and encryption key used between NCS 1004 and the RADIUS server.

Example:

```
RP/0/RP0/CPU0:ios(config-radius-host)# key 7 08984B1A4D0C19157A5F57
```

This key overrides the global setting of the **radius-server key** command. If you do not specify a key string, the global value is used.

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. This is because the leading spaces are ignored, but spaces within and at the end of the key are used. If you use spaces in the key, do not enclose the key in quotation marks unless the quotation marks themselves are part of the key.

Step 7 Repeat steps 3 through 6 for each external RADIUS server that you want to add to the server group.

What to do next

After configuring the RADIUS server parameters, enable AAA authentication for the RADIUS server group. For details, see Enable RADIUS authentication.

Enable RADIUS authentication

Enable AAA for the RADIUS server group and verify the configuration.

Before you begin

Before you begin, ensure that you have configured RADIUS server parameters. For details, see Configure RADIUS server parameters.

Follow these steps to enable RADIUS authentication:

Procedure

Step 1 Enter the **aaa authentication login** command to specify the default method list for authentication and enable authentication for console in global configuration mode.

Example:

```
RP/0/RP0/CPU0:ios(config-radius-host)# aaa authentication login default group radius local
```

Step 2 Commit the configuration changes.

Example:

```
RP/0/RP0/CPU0:ios(config-radius-host)# commit
```

Alternatively, you can use the **end** command to commit and exit configuration mode.

Step 3 (Optional) Enter the **show radius server-groups** command to display information about each configured RADIUS server group in the system.

Example:

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

TACACS+ security protocol

TACACS+ is a security protocol designed to provide centralized authentication, authorization, and accounting services for network devices. The protocol offers:

- centralized user validation for enhanced security,
- detailed accounting information for monitoring user activities, and
- flexible administrative control over user access and permissions.

TACACS+ authentication flow on NCS 1004

The TACACS+ application is designed to enhance the security of the NCS 1004 device by centralizing user validation. It uses AAA commands and can be enabled and configured on NCS 1004 for improved security.

When the TACACS+ server is configured and protocol is enabled on the node, user credentials are authenticated through the TACACS+ server. When the user attempts to log into the node, the user name and password are forwarded to the configured TACACS+ servers to obtain authentication status.

If authentication fails through the TACACS+ server, the credentials are sent to the node and authenticated locally. If the authentication fails against the node, the user is not allowed to log into the node.

Configure TACACS+ server groups

TACACS+ server groups help organize existing server hosts into logical groups. This allows you to select a subset of preconfigured server hosts for a particular service.

Before you begin

For successful configuration, the external server must be accessible at the time of configuration. If you assign the same IP address for global configuration, you must specify server-private parameters.

Configuring NCS 1004 to use AAA server groups allows you to group existing server hosts. The system uses a server group along with a global server host list. The server group includes the IP addresses of selected server hosts.

You can enter one or more **server** commands. The **server** command specifies the hostname or IP address of an external TACACS+ server. After configuration, you can reference this server group from the AAA method lists for authentication, authorization, or accounting.

To configure TACACS+ server groups, complete these subtasks:

Procedure

Step 1 Configure TACACS+ server parameters. For details, see Configure TACACS+ server parameters.

Step 2 Enable AAA authentication and authorization for the TACACS+ server group. For details, see Enable TACACS+ authentication and authorization.

Configure TACACS+ server parameters

This task configures the basic TACACS+ server group and adds external TACACS+ servers with their IP addresses, port numbers, encryption keys, and timeout values.

Procedure

Step 1 Enter the IOS XR configuration mode.

Example:

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

Step 2 Enter the **aaa group server tacacs+** command to create an AAA server-group and access the server-group sub-configuration mode.

Example:

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

Step 3 Enter the **server-private** command to configure the IP address of the private TACACS+ server for the group server.

Example:

```
RP/0/RP0/CPU0:ios(config-sg-tacacs+)# server-private 10.1.1.1 port 49 key a_secret
```

Note

- You can configure up to ten TACACS+ private servers in a server group.
- If you do not specify private server parameters, the system uses global configurations. If you do not specify global configurations, the system uses default values.

Step 4 Enter the **key** command to configure the authentication and encryption key used between NCS 1004 and the TACACS+ daemon running on the TACACS+ server.

Example:

```
RP/0/RP0/CPU0:ios(config-sg-tacacs+)# key 7 08984B1A4D0C19157A5F57
```

If no key string is specified, the global value is used.

Step 5 Enter the **timeout** command to configure the timeout value that sets the length of time the AAA server waits to receive a response from the TACACS+ server.

Example:

```
RP/0/RP0/CPU0:ios(config-sg-tacacs-private)# timeout 4
```

Step 6 Repeat steps 3 through 5 to add additional servers to the server group.

What to do next

After configuring the TACACS+ server parameters, enable AAA authentication and authorization for the TACACS+ server group. For details, see Enable TACACS+ authentication and authorization.

Enable TACACS+ authentication and authorization

Enable AAA for the TACACS+ server group and verify the configuration.

Before you begin

Ensure that you have configured the TACACS+ server parameters. For details, see Configure TACACS+ server parameters.

Procedure

Step 1 Enter the **aaa authorization exec** command to configure certificate-based authentication for users in TACACS+ server groups.

Example:

```
RP/0/RP0/CPU0:ios(config-sg-tacacs-private)# aaa authorization exec default group TACACS_ALL
 local
```

Step 2 Enter the **aaa authorization login** command to set the default authentication method list and enable authentication for the console in global configuration mode.

Example:

```
RP/0/RP0/CPU0:ios(config-sg-tacacs-private)# aaa authentication login default group TACACS_ALL
 local
```

Step 3 Commit the changes and exit all configuration modes.

Example:

```
RP/0/RP0/CPU0:ios(config-sg-tacacs-private)# commit
RP/0/RP0/CPU0:ios(config-sg-tacacs-private)# end
```

Step 4 (Optional) Enter the **show tacacs server-groups** command to verify the TACACS+ server group configuration details.

Example:

```
RP/0/RP0/CPU0:ios# show tacacs server-groups
```

User activity accounting records are sent to the TACACS+ security server. This enables robust management, billing, and auditing.

Configure TACACS+ server

Enabling the AAA accounting feature on a switch allows it to track the network services that users are accessing and the amount of network resources they are using. The switch then sends this user activity data to the TACACS+ security server in the form of accounting records. Each record contains attribute-value pairs and is saved on the security server for analysis. This data can be used for network management, client billing, or auditing purposes.

Procedure

Step 1 Enter the IOS XR configuration mode.

Example:

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

Step 2 Enter the **aaa accounting exec** command to enable the TACACS+ accounting to send a start-record accounting notice at the beginning of a privileged EXEC process and a stop-record at the end.

Example:

```
RP/0/RP0/CPU0:ios(config)# aaa accounting exec default start-stop group TACACS_ALL
```

Step 3 Enter the **aaa accounting exec** command to create a default command accounting method list for accounting services provided by a TACACS+ security server.

Example:

```
RP/0/RP0/CPU0:ios(config)# aaa accounting exec default start-stop group TACACS_ALL
```

This list is configured for privilege level commands and set with a stop-only restriction.

The switch begins recording user access to network services and sends accounting records to the TACACS+ server for management, billing, or auditing.

Deprecation of Type 7 password and Type 5 secret

Password configuration options before Release 24.4.1:

Until Release 24.4.1, there were two options for configuring a password:

- Password: uses Type 7 encryption to store the password.
- Secret: supports Type 5, 8, 9, or 10 hashing algorithms to store the password securely.

Starting with Release 24.4.1, Type 7 password and Type 5 secret are deprecated because of security concerns. The deprecation process began in Release 24.4.1 and will be complete in a future release. We recommend using the default option, Type 10 secret.

password

The **password** options available in CLI from Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#password ?
LINE The type 7 password followed by '7 ' OR SHA512-based password (deprecated, use 'secret')
```

Changes:

- All the options that were present until Release 24.4.1 are removed except LINE (to accept cleartext).
- During upgrade: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.
- Post-upgrade: You can still use the Type 7 password option in new commits, but the password is stored as Type 10 secret.
- New syslog has been added to indicate the deprecation process:

  ```
  %SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is deprecated.

   Converting it to a Type 10 secret for user <user name>.
  ```

- **show running configuration** command output before upgrade:

  ```
  username example
  password 7 106D000A0618
  !
  ```

  show running configuration command output post-upgrade:

  ```
  username example
  secret 10
  $6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iClOryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqq8st.
  !
  ```

masked-password

The **masked-password** options available in CLI from Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#masked-password ?
0 Specifies a cleartext password will follow
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
<cr> The cleartext user password
```

Changes:

- The options 7 and encrypted, available in Release 24.4.1 and earlier, are removed.
- During upgrade: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.
- Post-upgrade: Masked-password remains an alternate method for configuring the password. When you use the masked-password keyword with a clear string in new commits, the password is stored as Type 10 secret.
- New syslog has been added to indicate the deprecation process:

  ```
  %SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is deprecated.

   Converting it to a Type 10 secret for user <user name>.
  ```

- **show running configuration** command output before upgrade:

```
username example
password 7 106D000A0618
!
```

show running configuration command output post-upgrade:

```
username example
secret 10
$6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iClOryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqq8st.
!
```

password-policy

The **password-policy** options available in CLI from Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#password-policy ?
WORD Specify the password policy name

RP/0/RP0/CPU0:ios(config-un)#password-policy abcd password ?
0 Specifies an UNENCRYPTED password will follow
7 Specifies that an encrypted password will follow
LINE The UNENCRYPTED (cleartext) user password
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
encrypted Config deprecated. Will be removed in 7.7.1. Specify '7' instead.
```

Changes:

- All the options that were present until 24.4.1 are removed except LINE (to accept cleartext).
- During upgrade: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.
- Post-upgrade: You can still use the password-policy option with new commits, but the password is stored as Type 10 secret.

aaa password-policy

The **aaa password-policy** options available in CLI from Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config)#aaa password-policy abcd
RP/0/RP0/CPU0:ios(config-pp)#?
min-char-change Number of characters change required between old and new passwords
(deprecated, will be removed in 25.3.1)
restrict-password-advanced Advanced restrictions on new password (deprecated, will be removed
 in 25.3.1)
restrict-password-reverse Restricts the password to be same as reversed old password
(deprecated, will be removed in 25.3.1)
```

Changes:

- The options min-char-change, restrict-password-advanced, and restrict-password-reverse, available in Release 24.4.1 and earlier, are now deprecated.
- During upgrade: These deprecated configurations do not go through any change during upgrade.
- Post-upgrade: These deprecated keywords do not take effect when configured post-upgrade.
- New syslog messages have been added to indicate the deprecation process:

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy option
'min-char-change' is deprecated.
Password/Secret will not be checked against this option now.
```

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy option
'restrict-password-reverse' is deprecated.
 Password/Secret will not be checked against this option now.

%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy option
'restrict-password-advanced' is deprecated.
Password/Secret will not be checked against this option now.
```

- **show running configuration** command output before upgrade:

```
aaa password-policy abcd
lower-case 3
min-char-change 1
restrict-password-reverse
restrict-password-advanced
!
```

show running configuration command output post-upgrade:

```
aaa password-policy abcd
lower-case 3
min-char-change 1
restrict-password-reverse
restrict-password-advanced
!
```

secret

The **secret** options available in CLI from Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#secret ?
0 Specifies a cleartext password will follow
10 Specifies that SHA512-based password will follow
8 Specifies that SHA256-based password will follow
9 Specifies that Scrypt-based password will follow
LINE The cleartext user password

RP/0/RP0/CPU0:ios(config-un)#secret 0 enc-type ?
<8-10> Specifies which algorithm to use. Only 8,9,10 supported [Note: Option '5' is not
available to use from 24.4]
```

Changes:

- The options 5 and encrypted are removed.
- During upgrade: Configurations using Type 5 secret will remain unchanged.
- Post-upgrade: Although keyword 5 is deprecated, you can still use existing configurations with Type 5 secret.
- New syslog has been added to indicate the deprecation process:

```
%SECURITY-LOCALD-2-DEPRECATED_SECRET_TYPE : Type 5 secret is deprecated.
Please use the 'secret' keyword with option type 10 for user.
```

- **show running configuration** command output before upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
!
!
```

show running configuration command output post-upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
```

```
!
!
```

masked-secret

The **masked-secret** options available in CLI from Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#masked-secret ?
0 Specifies a cleartext password will follow
10 Specifies that SHA512-based password will follow
8 Specifies that SHA256-based password will follow
9 Specifies that Scrypt-based password will follow
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
<cr> The cleartext user password
```

Changes:

- The options 5 and encrypted are removed.
- During upgrade: Configurations using masked-secret with Type 5 will remain unchanged.
- Post-upgrade: Although keyword 5 is deprecated, you can still use existing configurations with Type 5 masked secret.
- New syslog has been added to indicate the deprecation process:

  ```
  %SECURITY-LOCALD-2-DEPRECATED_SECRET_TYPE : Type 5 secret is deprecated.
  Please use the 'secret' keyword with option type 10 for user.
  ```

- **show running configuration** command output before upgrade:

  ```
  username example
  secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
  !
  !
  ```

  show running configuration command output post-upgrade:

  ```
  username example
  secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
  !
  !
  ```