

Configuring Authorization

AAA authorization enables you to limit the services available to a user. When AAA authorization is enabled, the network access server uses information retrieved from the user's profile, which is located either in the local user database or on the security server, to configure the user's session. Once this is done, the user will be granted access to a requested service only if the information in the user profile allows it.

This chapter describes the following topics and tasks:

- AAA Authorization Types
- Named Method Lists for Authorization
- AAA Authorization Methods
- AAA Authorization Prerequisites
- AAA Authorization Configuration
- Configure Authorization
- Configure AAA Authorization Using Named Method Lists
- Disable Authorization for Global Configuration Commands
- Authorization for Reverse Telnet
- Authorization Attribute-Value Pairs
- Authorization Configuration Examples

For a complete description of the authorization commands used in this chapter, refer to the "Authorization Commands" chapter in the *Security Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

AAA Authorization Types

Cisco IOS software supports three different types of authorization:

- **EXEC**—Applies to the attributes associated with a user EXEC terminal session.
- **Command**—Applies to the EXEC mode commands a user issues. Command authorization attempts authorization for all EXEC mode commands, including global configuration commands, associated with a specific privilege level.
- **Network**—Applies to network connection. This can include a PPP, SLIP, or ARAP connection.

Named Method Lists for Authorization

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 simply a named list describing the authorization methods to be queried (such as RADIUS or TACACS+), in sequence. Method lists enable you to designate one or more security protocols to be used for authorization, thus ensuring a backup system in case the initial method fails. Cisco IOS software uses the first method listed to authorize users for specific network services; if that method fails to respond, the Cisco IOS software selects the next method listed in the method list. This process continues until there is successful communication with a listed authorization method, or all methods defined are exhausted.

Note The Cisco IOS software attempts authorization with the next listed method only when there is no response 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.

Cisco IOS software supports the following six methods for authorization:

- **TACACS+**—The network access server exchanges authorization information with the TACACS+ security daemon. TACACS+ authorization defines specific rights for users by associating attribute-value (AV) pairs, which are stored in a database on the TACACS+ security server, with the appropriate user.
- **If-Authenticated**—The user is allowed to access the requested function provided the user has been authenticated successfully.
- **None**—The network access server does not request authorization information; authorization is not performed over this line/interface.
- **Local**—The router or access server consults its local database, as defined by the **username** command, to authorize specific rights for users. Only a limited set of functions can be controlled via the local database.
- **RADIUS**—The network access server requests authorization information from the RADIUS security server. RADIUS authorization defines specific rights for users by associating attributes, which are stored in a database on the RADIUS server, with the appropriate user.
- **Kerberos Instance Map**—The network access server uses the instance defined by the **kerberos instance map** command for authorization.

Method lists are specific to the type of authorization being requested. AAA supports four different types of authorization:

- **Network**—Applies to network connections. This can include a PPP, SLIP, or ARAP connection.
- **EXEC**—Applies to the attributes associated with a user EXEC terminal session.
- **Commands**—Applies to the EXEC mode commands a user issues. Command authorization attempts authorization for all EXEC mode commands, including global configuration commands, associated with a specific privilege level.
- **Reverse Access**—Applies to reverse Telnet sessions.

When you create a named method list, you are defining a particular list of authorization methods for the indicated authorization type.

Once defined, method lists must be applied to specific lines or interfaces before any of the defined methods will be performed. The only exception is the default method list (which is named “default”). If the **aaa authorization** command for a particular authorization type is issued without a named method list specified, the default method list is automatically applied to all interfaces or lines except those that have a named method list explicitly defined. (A defined method list overrides the default method list.) If no default method list is defined, then no authorization takes place.

AAA Authorization Methods

AAA supports five different methods of authorization:

- **TACACS+**—The network access server exchanges authorization information with the TACACS+ security daemon. TACACS+ authorization defines specific rights for users by associating attribute-value pairs, which are stored in a database on the TACACS+ security server, with the appropriate user.
- **If-Authenticated**—The user is allowed to access the requested function provided the user has been authenticated successfully.
- **Local**—The router or access server consults its local database, as defined by the **username** command, to authorize specific rights for users. Only a limited set of functions can be controlled via the local database.
- **RADIUS**—The network access server requests authorization information from the RADIUS security server. RADIUS authorization defines specific rights for users by associating attributes, which are stored in a database on the RADIUS server, with the appropriate user.
- **Kerberos Instance Map**—The network access server uses the instance defined by the **kerberos instance map** command for authorization.

AAA Authorization Prerequisites

Before configuring authorization using named method lists, you must first perform the following tasks:

- Enable AAA on your network access server. For more information about enabling AAA on your Cisco router or access server, refer to the “AAA Overview” chapter.
- Configure AAA authentication. Authorization generally takes place after authentication and relies on authentication to work properly. For more information about AAA authentication, refer to the “Configuring Authentication” chapter.
- Define the characteristics of your RADIUS or TACACS+ security server if you are issuing RADIUS or TACACS+ authorization. For more information about configuring your Cisco network access server to communicate with your RADIUS security server, refer to the “Configuring RADIUS” chapter. For more information about configuring your Cisco network access server to communicate with your TACACS+ security server, refer to the “Configuring TACACS+” chapter.

- Define the rights associated with specific users by using the **username** command if you are issuing local authorization. For more information about the **username** command, refer to the *Security Command Reference*.
- Create the administrative instances of users in the Kerberos key distribution center by issuing the **kerberos instance map** command if you are using Kerberos. For more information about Kerberos, refer to the “Configuring Kerberos” chapter.

AAA Authorization Configuration

This section describes the following:

- Configure Authorization
- Configure AAA Authorization Using Named Method Lists
- Disable Authorization for Global Configuration Commands
- Authorization for Reverse Telnet
- Authorization Attribute-Value Pairs

For authorization configuration examples using the commands in this chapter, refer to the “TACACS+ Configuration Examples” section located at the end of the this chapter.

Configure Authorization

The **aaa authorization** command allows you to set parameters that restrict a user’s network access. To enable AAA authorization, use the following command in global configuration mode:

Command	Purpose
aaa authorization { network exec command level } { tacacs+ if-authenticated none local radius krb5-instance }	Set parameters that restrict a user’s network access.

Note Authorization is bypassed for authenticated users who log in using the console line, even if authorization has been configured.

To enable authorization for all network-related service requests (including SLIP, PPP, PPP NCPs, and ARA protocols), use the **network** keyword. To enable authorization to determine if a user is allowed to run an EXEC shell, use the **exec** keyword.

To enable authorization for specific, individual EXEC commands associated with a specific privilege level, use the **command** keyword. This allows you to authorize all commands associated with a specified command level from 0 to 15.

TACACS+ Authorization

To have the network access server request authorization information via a TACACS+ security server, use the **aaa authorization** command with the **tacacs+ method** keyword. For more specific information about configuring authorization using a TACACS+ security server, refer to the “Configuring TACACS+” chapter. For an example of how to enable a TACACS+ server to authorize the use of network services, including PPP and ARA, see the “TACACS+ Authorization Example” section at the end of this chapter.

If-Authenticated Authorization

To allow users to have access to the functions they request as long as they have been authenticated, use the **aaa authorization** command with the **if-authenticated** *method* keyword. If you select this method, all requested functions are automatically granted to authenticated users.

None Authorization

To perform no authorization for the actions associated with a particular type of authentication, use the **aaa authorization** command with the **none** *method* keyword. If you select this method, authorization is disabled for all actions.

Local Authorization

To select local authorization, which means that the router or access server consult its local user database to determine the functions a user is permitted, use the **aaa authorization** command with the **local** *method* keyword. The functions associated with local authorization are defined by using the **username** global configuration command. For a list of permitted functions, refer to the “Configuring Authentication” chapter.

RADIUS Authorization

To have the network access server request authorization via a RADIUS security server, use the **aaa authorization** command with the **radius** *method* keyword. For more specific information about configuring authorization using a RADIUS security server, refer to the “Configuring RADIUS” chapter. For an example of how to enable a RADIUS server to authorize services, see the “RADIUS Authorization Example” section at the end of this chapter.

Kerberos Authorization

To run authorization to determine if a user is allowed to run an EXEC shell at a specific privilege level based on a mapped Kerberos instance, use the **krb5-instance** *method* keyword. For more information, refer to the “Enable Kerberos Instance Mapping” section of the “Configuring Kerberos” chapter. For an example of how to enable Kerberos instance mapping, see the “Kerberos Instance Mapping Examples” section at the end of this chapter.

Configure AAA Authorization Using Named Method Lists

To configure AAA authorization using named method lists, use the following commands beginning in global configuration mode:

Step	Command	Purpose
1	aaa authorization { network exec commands level reverse-access } { default <i>list-name</i> } [<i>method1</i> [<i>method2...</i>]]	Create an authorization method list for a particular authorization type and enable authorization.
2	line [aux console tty vty] <i>line-number</i> [<i>ending-line-number</i>] interface <i>interface-type interface-number</i>	Enter the line configuration mode for the lines to which you want to apply the authorization method list. or Enter the interface configuration mode for the interfaces to which you want to apply the authorization method list.
3	authorization { arap commands level exec reverse-access } { default <i>list-name</i> } ppp authorization { default <i>list-name</i> }	Apply the authorization list to a line or set of lines. or Apply the authorization list to an interface or set of interfaces.

Authorization Types

Named authorization method lists are specific to the indicated type of authorization. To create a method list to enable authorization for all network-related service requests (including SLIP, PPP, PPP NCPs, and ARA protocols), use the **network** keyword.

To create a method list to enable authorization to determine if a user is allowed to run an EXEC shell, use the **exec** keyword.

To create a method list to enable authorization for specific, individual EXEC commands associated with a specific privilege level, use the **commands** keyword. (This allows you to authorize all commands associated with a specified command level from 0 to 15.)

To create a method list to enable authorization for reverse Telnet functions, use the **reverse-access** keyword.

For information about the types of authorization supported by the Cisco IOS software, refer to the “AAA Authorization Types” section.

Authorization Methods

To have the network access server request authorization information via a TACACS+ security server, use the **aaa authorization** command with the **tacacs+ method** keyword. For more specific information about configuring authorization using a TACACS+ security server, refer to the “Configuring TACACS+” chapter.

To allow users to have access to the functions they request as long as they have been authenticated, use the **aaa authorization {type}** command with the **if-authenticated method** keyword. If you select this method, all requested functions are automatically granted to authenticated users.

There may be times when you do not want to run authorization from a particular interface or line. To stop authorization activities on designated lines or interfaces, use the **none method** keyword.

To select local authorization, which means that the router or access server consults its local user database to determine the functions a user is permitted, use the **local method** keyword. The functions associated with local authorization are defined by using the **username** global configuration command. For a list of permitted functions, refer to the “Configuring Authentication” chapter.

To have the network access server request authorization via a RADIUS security server, use the **radius method** keyword. For more specific information about configuring authorization using a RADIUS security server, refer to the “Configuring RADIUS” chapter.

To run authorization to determine if a user is allowed to run an EXEC shell at a specific privilege level based on a mapped Kerberos instance, use the **krb5-instance method** keyword. For more information, refer to the “Enable Kerberos Instance Mapping” section of the “Configuring Kerberos” chapter.

Note Authorization method lists for SLIP follow whatever is configured for PPP on the relevant interface. If no lists are defined and applied to a particular interface (or no PPP settings are configured), the default setting for authorization applies.

Disable Authorization for Global Configuration Commands

The **aaa authorization** command with the keyword **command** attempts authorization for all EXEC mode commands, including global configuration commands, associated with a specific privilege level. Because there are configuration commands that are identical to some EXEC-level commands, there can be some confusion in the authorization process. Using **no aaa authorization config-commands** stops the network access server not from attempting configuration command authorization. To disable AAA authorization for all global configuration commands, use the following command in global configuration mode:

Command	Purpose
no aaa authorization config-command	Disable authorization for all global configuration commands.

Authorization for Reverse Telnet

Telnet is a standard terminal emulation protocol used for remote terminal connection. Normally, you log in to a network access server (typically through a dialup connection) and then use Telnet to access other network devices from that network access server. There are times, however, when it is necessary to establish a reverse Telnet session. In reverse Telnet sessions, the Telnet connection is established in the opposite direction—from inside a network to a network access server on the network periphery to gain access to modems or other devices connected to that network access server. Reverse Telnet is used to provide users with dialout capability by allowing them to Telnet to modem ports attached to a network access server.

It is important to control access to ports accessible through reverse Telnet. Failure to do so could, for example, allow unauthorized users free access to modems where they can trap and divert incoming calls or make outgoing calls to unauthorized destinations.

Authentication during reverse Telnet is performed through the standard AAA login procedure for Telnet. Typically the user has to provide a username and password to establish either a Telnet or reverse Telnet session. Reverse Telnet authorization provides an additional (optional) level of security by requiring authorization in addition to authentication. When enabled, reverse Telnet

authorization can use RADIUS or TACACS+ to authorize whether or not this user is allowed reverse Telnet access to specific asynchronous ports, after the user successfully authenticates through the standard Telnet login procedure.

Reverse Telnet authorization offers the following benefits:

- An additional level of protection by ensuring that users engaged in reverse Telnet activities are indeed authorized to access a specific asynchronous port using reverse Telnet.
- An alternative method (other than access lists) to manage reverse Telnet authorization.

To configure a network access server to request authorization information from a TACACS+ or RADIUS server before allowing a user to establish a reverse Telnet session, use the following command in global configuration mode:

Command	Purpose
<code>aaa authorization reverse-access {radius tacacs+}</code>	Configure the network access server to request authorization information before allowing a user to establish a reverse Telnet session.

This feature enables the network access server to request reverse Telnet authorization information from the security server, whether RADIUS or TACACS+. You must configure the specific reverse Telnet privileges for the user on the security server itself.

Authorization Attribute-Value Pairs

RADIUS and TACACS+ authorization both define specific rights for users by processing attributes, which are stored in a database on the security server. For both RADIUS and TACACS+, attributes are defined on the security server, associated with the user, and sent to the network access server where they are applied to the user's connection.

For a list of supported RADIUS attributes, refer to the "RADIUS Attributes" appendix. For a list of supported TACACS+ AV pairs, refer to the "TACACS+ Attribute-Value Pairs" appendix.

Authorization Configuration Examples

This section contains the following configuration examples:

- Named Method List Configuration Example
- TACACS+ Authorization Examples
- RADIUS Authorization Example
- Kerberos Instance Mapping Examples
- Reverse Telnet Authorization Examples

Named Method List Configuration Example

The following example configures a Cisco AS5200 (enabled for AAA and communication with a RADIUS security server) for AAA services to be provided by the RADIUS server. If the RADIUS server fails to respond, then the local database will be queried for authentication and authorization information, and accounting services will be handled by a TACACS+ server.

```
aaa new-model
aaa authentication login admins local
aaa authentication ppp dialins radius local
aaa authorization network scoobee radius local
aaa accounting network charley start-stop radius

username root password ALongPassword

radius-server host alcatraz
radius-server key myRaDiUSpassWoRd

interface group-async 1
  group-range 1 16
  encapsulation ppp
  ppp authentication chap dialins
  ppp authorization scoobee
  ppp accounting charley

line 1 16
  autoselect ppp
  autoselect during-login
  login authentication admins
  modem dialin
```

The lines in this sample RADIUS AAA configuration are defined as follows:

- The **aaa new-model** command enables AAA network security services.
- The **aaa authentication login admins local** command defines a method list, admins, for login authentication.
- The **aaa authentication ppp dialins radius local** command defines the authentication method list “dialins,” which specifies that RADIUS authentication then (if the RADIUS server does not respond) local authentication will be used on serial lines using PPP.
- The **aaa authorization network scoobee radius local** command defines the network authorization method list named scoobee, which specifies that RADIUS authorization will be used on serial lines using PPP. If the RADIUS server fails to respond, then local network authorization will be performed.
- The **aaa accounting network charley start-stop radius** command defines the network accounting method list named charley, which specifies that RADIUS accounting services (in this case, start and stop records for specific events) will be used on serial lines using PPP.
- The **username** command defines the username and password to be used for the PPP Password Authentication Protocol (PAP) caller identification.
- The **radius-server host** command defines the name of the RADIUS server host.
- The **radius-server key** command defines the shared secret text string between the network access server and the RADIUS server host.
- The **interface group-async** command selects and defines an asynchronous interface group.
- The **group-range** command defines the member asynchronous interfaces in the interface group.

- The **encapsulation ppp** command sets PPP as the encapsulation method used on the specified interfaces.
- The **ppp authentication chap dialins** command selects Challenge Handshake Authentication Protocol (CHAP) as the method of PPP authentication and applies the “dialins” method list to the specified interfaces.
- The **ppp authorization scoobee** command applies the scoobee network authorization method list to the specified interfaces.
- The **ppp accounting charley** command applies the charley network accounting method list to the specified interfaces.
- The **line** command switches the configuration mode from global configuration to line configuration and identifies the specific lines being configured.
- The **autoselect ppp** command configures the Cisco IOS software to allow a PPP session to start up automatically on these selected lines.
- The **autoselect during-login** command is used to display the username and password prompt without pressing the Return key. After the user logs in, the autoselect function (in this case, PPP) begins.
- The **login authentication admins** command applies the admins method list for login authentication.
- The **modem dialin** command configures modems attached to the selected lines to only accept incoming calls.

TACACS+ Authorization Examples

The following example uses a TACACS+ server to authorize the use of network services, including PPP and ARA. If the TACACS+ server is not available or an error occurs during the authorization process, the fallback method (none) is to grant all authorization requests:

```
aaa authorization network tacacs+ none
```

The following example allows network authorization using TACACS+:

```
aaa authorization network tacacs+
```

The following example provides the same authorization, but also creates address pools called *mci* and *att*:

```
aaa authorization network tacacs+
ip address-pool local
ip local-pool mci 172.16.0.1 172.16.0.255
ip local-pool att 172.17.0.1 172.17.0.255
```

These address pools can then be selected by the TACACS daemon. A sample configuration of the daemon follows:

```

user = mci_customer1 {
    login = cleartext "some password"
    service = ppp protocol = ip {
        addr-pool=mci
    }
}

user = att_customer1 {
    login = cleartext "some other password"
    service = ppp protocol = ip {
        addr-pool=att
    }
}

```

RADIUS Authorization Example

The following example shows how to configure the router to authorize using RADIUS:

```

aaa authorization exec radius if-authenticated
aaa authorization network radius

```

The lines in this sample RADIUS authorization configuration are defined as follows:

- The **aaa authorization exec radius if-authenticated** command configures the network access server to contact the RADIUS server to determine if users are permitted to start an EXEC shell when they log in. If an error occurs when the network access server contacts the RADIUS server, the fallback method is to permit the CLI to start, provided the user has been properly authenticated.

The RADIUS information returned may be used to specify an autocommand or a connection access list be applied to this connection.

- The **aaa authorization network radius** command configures network authorization via RADIUS. This can be used to govern address assignment, the application of access lists, and various other per-user quantities.

Note Since no fallback method is specified in this example, authorization will fail if, for any reason, there is no response from the RADIUS server.

Kerberos Instance Mapping Examples

The following global configuration example maps the Kerberos instance, *admin*, to enable mode:

```

kerberos instance map admin 15

```

The following example configures the router to check users' Kerberos instances and set appropriate privilege levels:

```

aaa authorization exec krb5-instance

```

For more information about configuring Kerberos, refer to the “Configuring Kerberos” chapter.

Reverse Telnet Authorization Examples

The following example causes the network access server to request authorization information from a TACACS+ security server before allowing a user to establish a reverse Telnet session:

```
aaa new-model
aaa authentication login default tacacs+
aaa authorization reverse-access tacacs+
!
tacacs-server host 172.31.255.0
tacacs-server timeout 90
tacacs-server key goaway
```

The lines in this sample TACACS+ reverse Telnet authorization configuration are defined as follows:

- The **aaa new-model** command enables AAA.
- The **aaa authentication login default tacacs+** command specifies TACACS+ as the default method for user authentication during login.
- The **aaa authorization reverse-access tacacs+** specifies TACACS+ as the method for user authorization when trying to establish a reverse Telnet session.
- The **tacacs-server host** command identifies the TACACS+ server.
- The **tacacs-server timeout** command sets the interval of time that the network access server waits for the TACACS+ server to reply.
- The **tacacs-server key** command defines the encryption key used for all TACACS+ communications between the network access server and the TACACS+ daemon.

The following example configures a generic TACACS+ server to grant a user, jim, reverse Telnet access to port tty2 on the network access server named godzilla and to port tty5 on the network access server named gamera:

```
user = jim
login = cleartext lab
service = raccess {
port#1 = godzilla/tty2
port#2 = gamera/tty5
```

Note In this example, “godzilla” and “gamera” are the configured host names of network access servers, not DNS names or alias.

The following example configures the TACACS+ server (CiscoSecure) to grant a user named jim reverse Telnet access:

```
user = jim
profile_id = 90
profile_cycle = 1
member = Tacacs_Users
service=shell {
default cmd=permit
}
service=raccess {
allow "c2511e0" "tty1" ".*"
refuse ".*" ".*" ".*"
password = clear "goaway"
```

Note CiscoSecure only supports reverse Telnet using the command line interface in versions 2.1(x) through version 2.2(1).

An empty “service=raccess { }” clause permits a user to have unconditional access to network access server ports for reverse Telnet. If no “service=raccess” clause exists, the user is denied access to any port for reverse Telnet.

For more information about configuring TACACS+, refer to the “Configuring TACACS+” chapter. For more information about configuring CiscoSecure, refer to the *Secure Access Control Server User Guide*, version 2.1(2) or greater.

The following example causes the network access server to request authorization from a RADIUS security server before allowing a user to establish a reverse Telnet session:

```
aaa new-model
aaa authentication login default radius
aaa authorization reverse-access radius
!
radius-server host 172.31.255.0
radius-server key go away
```

The lines in this sample RADIUS reverse Telnet authorization configuration are defined as follows:

- The **aaa new-model** command enables AAA.
- The **aaa authentication login default radius** command specifies RADIUS as the default method for user authentication during login.
- The **aaa authorization reverse-access radius** specifies RADIUS as the method for user authorization when trying to establish a reverse Telnet session.
- The **radius-server host** command identifies the RADIUS server.
- The **radius-server key** command defines the encryption key used for all RADIUS communications between the network access server and the RADIUS daemon.

The following example configures the RADIUS server to grant a user named “jim” reverse Telnet access at port tty2 on the network access server named godzilla:

```
Password = "goaway"
User-Service-Type = Shell-User
cisco-avpair = "raccess:port#1=godzilla/tty2"
```

The syntax "raccess:port=any/any" permits a user to have unconditional access to network access server ports for reverse Telnet. If no "raccess:port={nasname}/{tty number}" clause exists in the user profile, the user is denied access to reverse Telnet on all ports.

For more information about configuring RADIUS, refer to the “Configuring RADIUS” chapter.

