Testing the RADIUS Server

This chapter describes how to use **radclient**, a RADIUS server test tool you run from the command line to test your Cisco Prime Access Registrar RADIUS server. You can use **radclient** to create packets, send them to a specific server, and examine the response.

Because the **radclient** command is Tcl-based, you can use it interactively or you can execute it as a Tcl script file.

To run the **radclient** command, enter:

radclient

After you enter the **radclient** command, you must log into the RADIUS server and provide an administrator's username, such as admin, and the administrator's password.

This chapter contains the following sections:

- radclient Command Syntax
- Working with Packets
- Attributes
- Using radclient Test Commands

radclient Command Syntax

The **radclient** command syntax is:

radclient [**-C** <*clustername*>] [**-N** <*adminname*>] [**-P** <*adminpassword*>] [**-i**] [**-n**] [**-p** <*load_path*>] **[-v] [-z debug_flags]** [**-I** flag]

Valid flags are:

- **-C** <*clustername*>
- **-N** <*adminname*>
- **-P** <*adminpassword*>
- **-i**—Forces interactive mode
- **-n**—Skips loading **radclient.tcl**
- **-p <path>**—Specifies the load_path
- **-s**—Uses default cluster, admin user, and password

  If you delete the admin user or modify the admin user's password, this option will no longer work.

- **-S <file>**—Sources specified file
- -**v**—Prints version and exits
- -**I** *<0 or 1>*—Enables to set as IPv4 or IPv6 client. *0* specifies IPv4 client and *1* specifies IPv6 client

  -z debug_flags—Specify debug levels. Debug flags must be of the format *X*=*n*, where *X* is the letter corresponding to the type of debug information you want to see, and *n* is the value. The higher the value, the more output. *X* can also be a string or a range of letters.

  For example, the following command line sets the debug levels for A, B, and C to 3:

  radclient -z ABC=3

  The following example command line sets the debug levels for everything between A and Z inclusive and l to 5:

  radclient -z A-Zl=5

Working with Packets

Using the **radclient** command, you can create packets (default or specific packets), view packets, send packets, read the value of packets, and delete packets.

This section contains the following topics:

- Creating Packets
- Creating CHAP Access-Request Packets
- Viewing Packets
- Sending Packets
- Creating Empty Packets
- Setting Packet Fields
- Reading Packet Fields
- Deleting Packets

Creating Packets

To create a basic RADIUS Access-Request packet, use the **radclient** command **simple**. This function creates a packet and fills in basic attributes. The syntax of the **simple** command is:

simple *<user_name> <user_password>*

For example, to create an Access-Request packet for user **bob** whose password is **bigDog**, enter:

simple bob bigDog

```
p001
```

The **radclient** command responds with `p001`, which is the identifier (name) of the newly created packet.

Creating CHAP Access-Request Packets

To create a CHAP Access-Request packet, use the **radclient** command **simple_chap**. The syntax of the **simple_chap** command is:

simple_chap <*user_name*> <*user_password*> <*use_challenge*>

<*use_challenge*> is a boolean that indicates whether to use the **CHAP-Challenge** attribute.

For example, to create a CHAP packet and use a <*use_challenge*>, enter:

simple_chap bob bigDog 1

```
p002
```

Viewing Packets

To view a packet or any other object, enter the object identifier at the **radclient** prompt. For example, to display packet `p001`, enter:

p001

```
Packet: code=Access-Request,id=0,length=0, attributes =
User-Name = bob
User-Password = bigDog
NAS-Identifier = localhost
NAS-Port = 0
```

Sending Packets

To send a packet, specify the packet identifier and enter the word **send**.

p001 send

You can optionally specify the host and port to which to send the packet. The default host is **localhost**, and the default port is **1812**.

When you want to send a packet to a different host and different port, you must specify them on the command line. For example, to send a packet to the RADIUS server `amazon`, at port number `1813`, enter:

p001 send amazon 1813

```
p002
```

When Prime Access Registrar receives a response to the packet you sent, it prints the response packet's object identifier before the **radclient** prompt returns.

The TCL variable *tries* determines how many times **radclient** retries to send the packet.

Creating Empty Packets

You can use **radclient** to **c**reate empty packets, them modify the packets to contain the appropriate fields. To create an empty packet, the syntax is:

packet <*packet-type*>

The optional <*packet-type*> argument can be the numerical RADIUS packet type identifier, such as `2`, or the string representation, such as `Access-Accept`:

packet 2

```
p00d
```

p00d

```
Packet: code = Access-Accept, id = 0, length = 0, attributes =
```

Setting Packet Fields

You can modify the value of a packet field using the following syntax:

<*packet-identifier*> **set** <*field*> <*value*>

<*packet-identifier*> is the packet number, such as `p001`.

<*field* > is the packet field you want to modify and can be one of the following:

- attrib—Set attributes in the packet; <*value*> is the attribute identifier.
- code— The packet type (such as Access-Request); <*value*> is either a numeric packet-type or the string representation (for example, 1 or Access Request).
- identifier— Set the packet ID; <*value*> is the numeric ID.
- length—Set the packet length; <*value*> is the numeric length.
- requestAuthenticator—Set the request authenticator; <*value*> is a hex string with a colon separating each byte.

<*value*> is either a numeric packet-type, the string representation, or the hex string with a colon separating each byte.

For example, to set the identifier field to `99`, enter:

p001 set identifier 99

```
99
```

p001

```
Packet: code = Access-Request, id = 99, length = 0, attributes =
        User-Name = bob
        User-Password = bigDog
        NAS-Identifier = localhost
        NAS-Port = 0
```

Reading Packet Fields

You can read (**get**) the value of any of the packet fields by using the syntax:

<*packet-identifier*> **get** <*attrib*>

For example, to **get** the **identifier** field, enter:

p001 get identifier

```
99
```

Deleting Packets

When you are writing long-running or iterating scripts, you might want to conserve memory by deleting packets when you are finished with them.

To delete a packet, enter:

<*packet-identifier*> **delete**

To delete all resources referred to by the packet `p001`, enter:

p001 delete

Attributes

Using the **radclient** command you can create specific RFC-defined attributes of requests and responses.

This section contains the following topics:

- Creating Attributes
- Setting Multivalued Attributes
- Viewing Attributes
- Getting Attribute Information
- Deleting Attributes
- Using the radclient Command

Creating Attributes

To create an attribute object, the syntax is:

<*attrib*> **name** <*value*>

<*attrib*> is a recognized RADIUS attribute name. <*value*> is the value of the attribute.

For example, to create the attribute **User-Name** and set its value to `bob`, enter:

attrib User-Name bob

```
a001
```

Note **a001** is the object identifier for the newly created attribute.

Setting Multivalued Attributes

Prime Access Registrar supports setting multivalued attributes (MVAs) in **radclient**. Use the set **mattrib** command to set multivalued attributes, as shown in the following example:

simple bob bob

```
p001
```

attrib cisco-avpair blah

```
a005
```

attrib cisco-avpair boo

```
a006
```

p001 set mattrib a005

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
User-Name = bob
User-Password = bob
NAS-Identifier = localhost
NAS-Port = 1
Cisco-AVPair = blah
```

p001 set mattrib a006

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
User-Name = bob
User-Password = bob
NAS-Identifier = localhost
NAS-Port = 1
Cisco-AVPair = blah
Cisco-AVPair = boo
```

Viewing Attributes

To view an attribute, or any other object, type the object identifier at the **radclient** prompt. For example, to display attribute `a001` created in the example above, enter:

a001

```
User-Name = bob
```

Getting Attribute Information

You can get the name and value of an attribute in various formats:

- get name—gets the name as a string
- get value—gets the value as a string
- get type—gets the name as an integer
- get valueAsInt—gets the value as an integer
- get valueAsIPAddress—gets the value as an IP address.

The following examples show how to get an attribute's name, type, value, and value as integer:

a001 get name

```
User-Name
```

a001 get type

```
1
```

a001 get value

```
bob
```

a001 get valueAsInt

```
a001: the value is not an int
```

Deleting Attributes

When you are writing long running or iterating scripts, you might want to conserve memory by deleting attributes when you are finished with them (be sure not to delete attributes being referred to by other objects, like packets.)

To delete all resources referred to by the attribute `a001`, enter:

a001 delete

Using the radclient Command

The following three examples show how to use **radclient** to create, send, and modify packets.

Example 1

This example creates an Access-Request packet for user `jane` with password `jane`, and sends it to the default RADIUS server (**localhost**).

simple jane jane

```
p001
```

The command **simple jane jane** creates the packet; the packet object identifier is **p001**. When you enter the packet object identifier, **radclient** displays the contents of the packet.

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
    User-Name = jane
    User-Password = jane
    NAS-Identifier = localhost
    NAS-Port = 0
```

When you enter the packet identifier and the command **send**, **radclient** sends the packet to the RADIUS server and prints the response packet object identifier.

p001 send

```
p002
```

When you enter the packet object identifier of the response, **radclient** displays the contents of the response packet.

p002

```
Packet: code = Access-Accept, id = 1, length = 38, attributes =
Login-IP-Host = 204.253.96.3
Login-Service = Telnet
Login-TCP-Port = 541
```

Example 2

The following example creates a simple Access-Request packet, then adds other attributes to it.

simple jane jane

```
p003
```

The command line above shows creation of the packet `p003` using user-ID `jane` and password `jane`.

attrib Service-Type Framed

```
a00c
```

The line above shows creating the **Service-Type** attribute (with the object identifier `a00c`).

a00c

```
Service-Type = Framed
```

Entering the attribute object identifier `a00c` displays the attribute object.

p003 set attrib a00c

The line above adds the newly set attribute to the packet. The following line creates another attribute.

attrib NAS-Port 99

```
a00d
```

a00d

```
NAS-Port = 99
```

p003 set attrib a00d

The same steps add the **NAS-Port** attribute to the packet, and finally, the packet contents are displayed.

p003

```
Packet: code = Access-Request, id = 0, length = 0, attributes =

User-Name = jane
User-Password = jane
NAS-Identifier = localhost
Service-Type = Framed
NAS-Port = 99
```

Example 3

Example 3 performs the same tasks as Example 2 using the command substitution feature of Tcl which allows you to use the results of one command as an argument to another command. Square brackets invoke command substitution. The statement inside the brackets is evaluated, and the result is used in place of the bracketed command.

simple jane jane

```
p004
```

p004 set attrib [attrib Service-Type Framed]

p004 set attrib [attrib NAS-Port 99]

p004

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
        User-Name = jane
        User-Password = jane
        NAS-Identifier = localhost
        Service-Type = Framed
        NAS-Port = 99
```

Using radclient Test Commands

You can use the **radclient** commands **timetest** and **callsPerSecond** to test the RADIUS server.

This section contains the following topics:

- radclient Variables
- Using timetest
- Using callsPerSecond
- Additional radclient Variables

radclient Variables

You control how **timetest** and **callsPerSecond** work using **radclient** variables. To set a **radclient** variable, use the **set** command as follows:

set ***variable value***

Table 8-1 lists the **radclient** variables used in **timetest** and **callsPerSecond** and their description.

Table 8-1 radclient Variables

Variable	Description
host	Destination host to send the packets (default is localhost)
num_packets	Number of packets to send at once (default is 256)
num_users	Modulus for the username pattern (default is 10000)
port	Port where **radclient** sends access-request packets (default is 1812). Changing this port does not affect the accounting_port.
retry_timeout	Length of time to wait after a timeout occurs before retrying
secret	Shared secret configured on the RADIUS server for the client (default is secret)
timeout	Length of time to wait before a timeout occurs
tries	Number of times to attempt to send
UserNamePattern	Pattern of the usernames (default is user%d%%PPP)
UserPasswordPattern	Pattern of the user passwords (default is user%d)

Using timetest

The **timetest** command sends a number of requests to the RADIUS server then waits for a response. When a response arrives, **timetest** immediately sends another request. **timetest** can keep up to 256 requests outstanding all the time.

The syntax of the **timetest** command is:

timetest <*testtype*> [<*cycles*> [<*repetitions*> [<*starting user number*> [<*increment user number*>]]]]

Table 8-2 lists the applicable test types.

Table 8-2 Test Types

Test Type	Description
1	Access-Request
2	Access-Request + Accounting-Start + Accounting-Stop
3	Accounting-Start + Accounting-Stop
4	Ascend-IPA-Allocate + Ascend-IPA-Release
5	Access-Request + Ascend-IPA-Allocate + Ascend-IPA-Release
6	Access-Request + Ascend-IPA-Allocate + Accounting-Start + Ascend-IPA-Release + Accounting-Stop
7	Access-Request + USR-Resource-Free-Request
8	LEAP Identity + LEAP-Challenge Response + LEAP Challenge
9	LEAP Identity + LEAP-Challenge Response + LEAP Challenge + Accounting-Start + Accounting-Stop
10	Access-Request + Accounting-Start + Accounting-Stop with Home-Agent request
11	Access-Request + Accounting-Start + Accounting-Stop with ODAP request

Consider this **timetest** example with **radclient** variables set to the following:

host—1.1.1.2

port—1812

secret—cisco

UserNamePattern—user%d

UserPasswordPattern—puser%d

num_users—100,000

num_packets—128

In this example, **timetest** sends packets directly to the host at IP address 1.1.1.2 on port 1812 with a shared secret `cisco`. There are 100,000 users in the server's user database with the name pattern *user#* and password pattern *puser#*, where # ranges from 0-99,999, inclusive. The number of outstanding requests are limited to 128.

Before starting the timing test, **timetest** sends an Accounting-On packet to the AAA Server and waits for a response to make sure that any session management being performed on the AAA Server is reset before running the test. After a response is received, the **timetest** can begin.

Using callsPerSecond

The **callsPerSecond** command is a smart throttle that sends packets at a rate you set. If you set **callsPerSecond** to two transactions per second (TPS), **callsPerSecond** sends a packet every 0.5 seconds.

The syntax of the **callsPerSecond** command is:

callsPerSecond <*callsPerSecond*> <*testtype*> [<*cycles*> [<*repetitions*> [<*starting user number*> [<*increment user number*>]]]]

Additional radclient Variables

Table 8-3 lists additional **radclient** variables and their description.

Table 8-3 Additional radclient Variables

Variable	Description
accounting_port	Port where the RADIUS server sends accounting packets (default is 1813). **Note** Changing accounting_port value does not affect the authentication port.
host	Name of host where Prime Access Registrar is installed
ignore_signature_errs	Causes server to ignore signature in the response
load_path	Search path to load source files with user processes
NASIdentifier	Value to set NAS-Identifier attribute
NASIPAddress	Value to set NAS-IP-Address attribute
NASPort	Value to set NAS-Port attribute
num_packets	Number of packets to send at once (default is 256)
num_users	Modulus for the username pattern (default is 10000)
port	Port where **radclient** sends access-request packets (default is 1812). Changing this port does not affect the accounting_port.
retry_timeout	Length of time to wait before attempting a retry
secret	Shared secret configured on the RADIUS server for the client (default is secret)
tclDefaultLibrary	Tclsh default library
tcl_patchLevel	Tclsh version with patch level
tcl_pkgPath	Tclsh install path
tcl_traceExec	Tclsh boolean to activate tracing
tcl_platform	Tclsh platform array
tcl_version	Tclsh version
tries	Number of retry attempts
UserNamePattern	Pattern of the usernames (default is user%d%%PPP)
UserPasswordPattern	Pattern of the user passwords (default is user%d)
verbose	Verbose flag for Tclsh