Chapter 11

Configuring Accounting and Authentication Plug-Ins with the SRC CLI

This chapter describes how to configure authentication and accounting plug-ins, with the SRC CLI. It also describes how to configure global and default retailer event publishers.

You can also use the C-Web interface to configure authorization and accounting plug-ins. For more information, see *SRC-PE C-Web Interface Configuration Guide, Chapter 28, Configuring Accounting and Authentication Plug-Ins with the C-Web Interface*.

Topics in this chapter include:

Creating RADIUS Peers

RADIUS peers are instances of RADIUS servers. If you define multiple servers, the SAE uses them in cases of failover or as alternate routers for load-balancing purposes.

Each RADIUS plug-in requires a default peer. Configure a RADIUS peer before you configure the plug-in.

RADIUS peers are configured in the peer group for each RADIUS plug-in. Use the following configuration statements to configure a RADIUS peer:

```
shared sae configuration plug-ins name name radius-accounting peer-group name {
    server-address server-address;
    server-port server-port;
    secret secret;
}

shared sae configuration plug-ins name name radius-authentication peer-group name {
    server-address server-address;
    server-port server-port;
    secret secret;
}

shared sae configuration plug-ins name name custom-radius-accounting peer-group
name {
    server-address server-address;
    server-port server-port;
    secret secret;
}

shared sae configuration plug-ins name name custom-radius-authentication peer-group
name {
    server-address server-address;
    server-port server-port;
    secret secret;
}

shared sae configuration plug-ins name name flex-radius-accounting peer-group name {
    server-address server-address;
    server-port server-port;
    secret secret;
}

shared sae configuration plug-ins name name flex-radius-authentication peer-group
name {
    server-address server-address;
    server-port server-port;
    secret secret;
}
```

To create a RADIUS peer:

1. From configuration mode, access the RADIUS peer configuration for the plug-in that you are configuring. In this sample procedure, the RADIUS peer is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting peer-group peer1**

2. Configure the IP address of the RADIUS server to which the SAE sends accounting data.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting peer-group peer1]
   user@host# **set server-address** *server-address*

3. Configure the port used for RADIUS packets.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting peer-group peer1]
   user@host# **set server-port** *server-port*

4. Configure the password that is shared with the RADIUS server. You must configure the same password on the RADIUS server.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting peer-group peer1]
   user@host# **set secret** *secret*

5. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name basicRadius
radius-accounting peer-group peer1]
user@host# show
server-address 10.10.1.1;
server-port 1812;
secret ********;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*

Configuring Tracking Plug-Ins

This section shows how to configure the tracking plug-ins described in Table 16.

By default, the fileAcct plug-in instance tracks all subscriber and service sessions and writes all available attributes to a file. You can use this plug-in instance or create new one.

NOTE: When you use the NAS-Port attribute in tracking plug-ins, the SAE calculates the NAS-Port value based on the NAS-Port-Id value that it receives from the JUNOSe router. You can change the NAS-Port format in the JUNOSe software. However, because the SAE has no indication of which format is configured on the JUNOSe router, the calculation of the NAS-Port attribute is correct only if the router uses the default configuration.

Table 16: Tracking Plug-Ins

Plug-In	Description
Basic RADIUS accounting	Sends accounting information to an external RADIUS accounting server or a group of redundant servers. Java class name—net.juniper.smgt.sae.plugin.RadiusTrackingPluginEventListener
Custom RADIUS accounting	Provides customized functions that can also be found in the flexible RADIUS accounting plug-ins. Custom plug-ins are internal plug-ins that are designed to deliver better system performance than the flexible RADIUS plug-ins. You can extend this plug-in by using the RADIUS client library. Java class name—net.juniper.smgt.sae.plugin.CustomRadiusAccouting
Flat file accounting	Writes tracking information to a file in comma-separated format. Java class name—net.juniper.smgt.sae.plugin.FileTrackingPluginEventListener
Flexible RADIUS accounting	Performs the same functions as the basic RADIUS accounting plug-in, but also lets you customize RADIUS accounting packets that the SAE sends to RADIUS servers. You can specify which fields are included in RADIUS accounting packets and what information is contained in the fields. Java class name—net.juniper.smgt.sae.plugin.FlexibleRadiusTrackingPluginEventListener
PCMM record-keeping server plug-in	Sends accounting information to an external PCMM record-keeping server (RKS). See *Configuring PCMM Record-Keeping Server Plug-Ins with SRC CLI* in *SRC-PE Solutions Guide, Chapter 5, Configuring the SAE for a PCMM Environment with the SRC CLI*. Java class name—net.juniper.smgt.sae.plugin.RksEventListener
QoS profile tracking	Ensures that as a subscriber activates and deactivates services, the correct QoS profile is attached to the subscriber interface. See *SRC-PE Solutions Guide, Chapter 1, Managing Tiered and Premium Services with QoS on JUNOSe Routers with the SRC CLI*. Java class name—net.juniper.smgt.sae.plugin.qtp.QosProfileTrackingPluginEventListener

Configuring Flat File Accounting Plug-Ins

Flat file accounting plug-ins write information to a file in a comma-separated format. The SRC software has a default flat file accounting plug-in instance called fileAcct. The fileAcct instance logs all possible attributes for 24-hour periods in the file *var/acct/log*.

Another item that you can configure for flat files is the names of the headers that appear in the file. See *Configuring Headers for Flat File Accounting Plug-Ins* on page 142.

Use the following configuration statements to create flat-file accounting plug-in instances:

```
shared sae configuration plug-ins name name file-accounting {
    filename filename;
    template template;
    interval interval;
    fields [(status | nas-id | host | router-name | interface-name | interface-alias |
    interface-descr | port-id | user-ip-address | login-name | accounting-id | auth-user-id |
    if-radius-class | if-session-id | service-name | radius-class | event-time | session-id |
    terminate-cause | session-time | in-octets | out-octets | in-packets | out-packets |
    nas-ip | user-mac-address | service-session-name | service-session-tag | user-type |
    user-radius-class | user-session-id | primary-user-name | subscription-name |
    login-id | if-index | event-time-millisecond | nas-port | operational | user-inet-address
    | nas-inet-address | router-type | interface-speed)...];
}
```

To create flat-file accounting plug-ins:

1. From configuration mode, access the basic RADIUS accounting plug-in configuration. In this sample procedure, the plug-in called fileAcct is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name fileAcct file-accounting**

2. Configure the name and location of the file to which the SAE writes accounting information.

   [edit shared sae group west-region configuration plug-ins name fileAcct file-accounting]
   user@host# **set filename** *filename*

3. Configure the name of the template that defines header names for attributes listed in accounting files.

   [edit shared sae group west-region configuration plug-ins name fileAcct file-accounting]
   user@host# **set template** *template*

4. Configure the number of hours of information stored in each accounting file.

   [edit shared sae group west-region configuration plug-ins name fileAcct file-accounting]
   user@host# **set interval** *interval*

5. Configure the fields that you want to record in the accounting file.

   [edit shared sae group west-region configuration plug-ins name fileAcct file-accounting]
   user@host# **set fields** [(status | nas-id | host | router-name | interface-name | interface-alias | interface-descr | port-id | user-ip-address | login-name | accounting-id | auth-user-id | if-radius-class | if-session-id | service-name | radius-class | event-time | session-id | terminate-cause | session-time | in-octets | out-octets | in-packets | out-packets | nas-ip | user-mac-address | service-session-name | service-session-tag | user-type | user-radius-class | user-session-id | primary-user-name | subscription-name | login-id | if-index | event-time-millisecond | nas-port | operational | user-inet-address | nas-inet-address | router-type | interface-speed)...]

6. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name fileAcct
file-accounting]
user@host# show
filename var/acct/log;
template FileAccounting.std;
interval 24;
fields [ status nas-id host router-name interface-name interface-alias
interface-descr port-id user-inet-address login-name accounting-id
auth-user-id if-session-id service-name event-time session-id
terminate-cause session-time in-octets out-octets in-packets out-packets
nas-inet-address user-mac-address service-session-name service-session-tag
user-type user-session-id ];
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*

Configuring Headers for Flat File Accounting Plug-Ins

When the SAE writes data to a flat file, it writes into the first line the headers that identify the attributes in the file. For example, in the following accounting file, the first line lists headers for all attribute fields in the file, and the following lines list the actual data in each field:

Accounting Status,NAS ID,SSP Host,Router Name,Interface Name,Interface Alias,Interface Description,NAS port ID,User IP Address,User ID,User Accounting ID,User Authentication ID,INTF Radius Class,INTF,SessionId, Service Name,Radius Class,Timestamp,SessionId, Terminate Cause,Session Time,Input Octets,Output Octets,Input Packets,Output Packets,NAS IP,User Mac address,Service Session Name,Service Session Tag,User Session Type,User Session Radius Class,User Session ID

start,SSP.uelmo,uelmo,default@erx7_ssp57,FastEthernet1/1.1,,IP1/1.1,default@erx7_ssp57 FastEthernet1/1:65535, 10.10.10.20,pebbles@virneo.net,,,,erx fastEthernet 1/1:0001048619,Video-Gold,Video-Gold,Fri Jan 30 14:23:29 EDT 2004, VideoGold:null:1064946209182, 0,0,0,0,0,0, 10.10.7.17,,,,PPP,, pebbles:1064946144841

You can assign your own names to the headers that appear in the file. To do so, define the header names in a template, and then set up file accounting plug-in instances to use the template. The default template, FileAccounting.std, defines header names for all possible attributes. You can use the default template or create your own templates.

Use the following configuration statements to create a file accounting template:

shared sae configuration file-accounting-template *name* ...

shared sae configuration file-accounting-template *name* attributes (status | nas-id | host | router-name | interface-name | interface-alias | interface-descr | port-id | user-ip-address | login-name | accounting-id | auth-user-id | if-radius-class | if-session-id | service-name | radius-class | event-time | session-id | terminate-cause | session-time | in-octets | out-octets | in-packets | out-packets | nas-ip | user-mac-address | service-session-name | service-session-tag | user-type | user-radius-class | user-session-id | primary-user-name | subscription-name | login-id | if-index | event-time-millisecond | nas-port | operational | user-inet-address | nas-inet-address | router-type | interface-speed | service-bundle | user-dn | uid | domain | retailer-dn | password | service-scope | session-timeout | downstream-bandwidth | upstream-bandwidth | dhcp-packet | aggr-session-id | aggr-login-name | aggr-user-dn | aggr-user-inet-address | aggr-accounting-id | aggr-auth-user-id) {
    value;
}

To set up a file accounting template:

1. From configuration mode, access the file accounting template configuration. In this sample procedure, the template called std is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration file-accounting-template std**

2. Define header names.

   [edit shared sae group west-region configuration file-accounting-template std]
   user@host# **set attributes** *attribute value*

   For example:

   [edit shared sae group west-region configuration file-accounting-template std]
   user@host# **set attributes terminate-cause "RADIUS Termination Cause"**

3. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration file-accounting-template
std]
user@host# show
attributes {
  terminate-cause "RADIUS Termination Cause";
  service-session-name "Service Session Name";
}
```

Configuring Basic RADIUS Accounting Plug-Ins

You can use basic RADIUS accounting plug-ins to send accounting information to an external RADIUS accounting server or to a group of redundant servers. To communicate with nonredundant servers, you need to create multiple instances of the plug-in.

Use the following configuration statements to configure RADIUS accounting plug-ins:

```
shared sae configuration plug-ins name name radius-accounting {
    load-balancing-mode (failover | roundRobin);
    failback-timer failback-timer;
    nas-ip (SspIp | ErxIp);
    retry-interval retry-interval;
    maximum-queue-length maximum-queue-length;
    bind-address bind-address;
    udp-port udp-port;
    username (login-name | accounting-id | auth-user-name | manager-id);
    calling-station-id (mac | no);
    default-peer default-peer;
}
```

To set up basic RADIUS accounting plug-ins:

1. From configuration mode, access the basic RADIUS accounting plug-in configuration. In this sample procedure, the plug-in called basicRadius is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting**

2. Configure the mode for load-balancing RADIUS servers.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set load-balancing-mode** (failover | roundRobin)

3. Specify if and when the SAE attempts to fail back to the default peer.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set failback-timer** *failback-timer*

4. (Optional) Configure the value of the NAS-IP attribute.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set nas-ip** (SspIp | ErxIp)

5. Configure the time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set retry-interval** *retry-interval*

6. Configure the maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set maximum-queue-length** *maximum-queue-length*

7. (Optional) Configure the source IP address that the plug-in uses to communicate with the RADIUS server. If you do not specify an address, the global default address is used.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set bind-address** *bind-address*

8. (Optional) Configure the source UDP port or a range of source UDP ports used for communication with the RADIUS server. If you do not specify a UDP port, the global UDP port is used.

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set udp-port** *udp-port*

9. Configure the value of the User-Name attribute (RADIUS attribute [1]).

   [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
   user@host# **set username** (login-name | accounting-id | auth-user-name | manager-id)

10. Specify whether the SAE sends the MAC address of the subscriber in the Calling-Station-Id attribute.

    [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
    user@host# **set calling-station-id** (mac | no)

11. Configure the default peer, which is the RADIUS server to which the SAE sends packets for this plug-in.

    [edit shared sae group west-region configuration plug-ins name basicRadius radius-accounting]
    user@host# **set default-peer** *default-peer*

12. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name basicRadius
radius-accounting]
user@host# show
load-balancing-mode failover;
failback-timer -1;
retry-interval 3000;
maximum-queue-length 10000;
username login-name;
calling-station-id no;
default-peer peer1;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI*.
- For information about setting up default peers, see *Creating RADIUS Peers* on page 138.

Configuring Flexible RADIUS Accounting Plug-Ins

Flexible RADIUS accounting plug-ins provide the same features as basic RADIUS accounting plug-ins. In addition, they allow you to customize RADIUS accounting packets that the SAE sends to RADIUS servers. You can specify which fields are included in the RADIUS accounting packets and what information is contained in the fields.

Use the following configuration statements to configure flexible RADIUS accounting plug-ins:

```
shared sae configuration plug-ins name name flex-radius-accounting {
    load-balancing-mode (failover | roundRobin);
    failback-timer failback-timer;
    timeout timeout;
    retry-interval retry-interval;
    maximum-queue-length maximum-queue-length;
    bind-address bind-address;
    udp-port udp-port;
    error-handling (0 | 1);
    default-peer default-peer;
    template template;
}
```

To set up flexible RADIUS accounting plug-ins:

1. From configuration mode, access the flexible RADIUS accounting plug-in configuration. In this sample procedure, the plug-in called flexRadiusAct is configured in the west-region SAE group.

   ```
   user@host# edit shared sae group west-region configuration plug-ins name
   flexRadiusAct flex-radius-accounting
   ```

2. Configure the mode for load-balancing RADIUS servers.

   ```
   [edit shared sae group west-region configuration plug-ins name flexRadiusAct
   flex-radius-accounting]
   user@host# set load-balancing-mode (failover | roundRobin)
   ```

3. Specify if and when the SAE attempts to fail back to the default peer.

   ```
   [edit shared sae group west-region configuration plug-ins name flexRadiusAct
   flex-radius-accounting]
   user@host# set failback-timer failback-timer
   ```

4. (Optional) Configure the maximum time the SAE waits for a response from a RADIUS server.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
   user@host# **set timeout** *timeout*

5. Configure the time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
   user@host# **set retry-interval** *retry-interval*

6. Configure the maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
   user@host# **set maximum-queue-length** *maximum-queue-length*

7. (Optional) Configure the source IP address that the plug-in uses to communicate with the RADIUS server. If you do not specify an address, the global default address is used.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
   user@host# **set bind-address** *bind-address*

8. (Optional) Configure the source UDP port or a range of source UDP ports used for communication with the RADIUS server. If you do not specify a UDP port, the global UDP port is used.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
   user@host# **set udp-port** *udp-port*

9. Configure the way the SAE handles errors.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
   user@host# **set error-handling** (0 | 1)

10. Configure the name of the RADIUS server to which the SAE sends packets for this plug-in.

    [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
    user@host# **set default-peer** *default-peer*

11. Configure the name of the RADIUS packet template that defines attributes for this plug-in.

    [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
    user@host# **set template** *template*

12. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name flexRadiusAct
flex-radius-accounting]
user@host# show
load-balancing-mode failover;
failback-timer -1;
timeout 15000;
retry-interval 3000;
maximum-queue-length 10000;
error-handling 0;
default-peer peer2;
template stdAcct;
peer-group peer2 {
  server-address 10.10.1.1;
  server-port 1818;
  secret ********;
}
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI*.
- For information about setting up default peers, see *Creating RADIUS Peers* on page 138.
- For information about defining RADIUS packet templates, see *Defining RADIUS Packets for Flexible RADIUS Plug-Ins with the SRC CLI* on page 163.

Configuring Custom RADIUS Accounting-Plug-Ins

The custom RADIUS accounting plug-ins provide the same functions as the flexible RADIUS accounting plug-ins, but are designed to deliver better system performance. To use a custom plug-in, you must provide a Java class that implements the service provider interface (SPI) defined in the RADIUS client library. Use this SPI to specify which fields and field values to include in RADIUS accounting packets. The RADIUS client library is part of the SAE core application programming interface (API).

See the documentation for the RADIUS client library in the SRC software distribution in the folder *SDK/doc/sae/net/juniper/smgt/sae/radiuslib* or in the SAE core API documentation on the Juniper Networks Web site at

http://www.juniper.net/techpubs/software/management/sdx/api-index.html

For a sample implementation, see the following directory in the SRC software distribution:

SDK/plugin/java/src/net/juniper/smgt/sample/radiuslib/RadiusPacketHandlerImpl.java.

Use the following configuration statements to set up custom RADIUS accounting plug-ins:

shared sae configuration plug-ins name *name* custom-radius-accounting {

```
    java-class-radius-packet-handler java-class-radius-packet-handler;
    class-path-radius-packet-handler class-path-radius-packet-handler;
    append-acct-status-type-attribute;
    require-mandatory-attributes;
    load-balancing-mode (failover | roundRobin);
    failback-timer failback-timer;
    timeout timeout;
    retry-interval retry-interval;
    maximum-queue-length maximum-queue-length;
    bind-address bind-address;
    udp-port udp-port;
    default-peer default-peer;
}
```

To set up custom RADIUS accounting plug-ins:

1. From configuration mode, access the custom RADIUS accounting plug-in configuration. In this sample procedure, the plug-in called customRadiusAct is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting**

2. Configure the name of the Java class that implements the RadiusPacketHandler interface in the RADIUS client library.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set java-class-radius-packet-handler** *java-class-radius-packet-handler*

3. Configure the URLs that identify a location from which Java classes are loaded when the plug-in is initialized.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting
   user@host# **set class-path-radius-packet-handler** *class-path-radius-packet-handler*

4. (Optional) Enable the plug-in to include the Acct-Status-Type attribute in a RADIUS accounting request packet.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set append-acct-status-type-attribute**

5. (Optional) Specify that a RADIUS authentication or accounting request must contain all mandatory RADIUS attributes before sending the request packet.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set require-mandatory-attributes**

6. Configure the mode for load-balancing RADIUS servers.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set load-balancing-mode** (failover | roundRobin)

7. Specify if and when the SAE attempts to fail back to the default peer.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set failback-timer** *failback-timer*

8. (Optional) Configure the maximum time the SAE waits for a response from a RADIUS server.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set timeout** *timeout*

9. Configure the time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet.

   [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
   user@host# **set retry-interval** *retry-interval*

10. Configure the maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.

    [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
    user@host# **set maximum-queue-length** *maximum-queue-length*

11. (Optional) Configure the source IP address that the plug-in uses to communicate with the RADIUS server. If you do not specify an address, the global default address is used.

    [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
    user@host# **set bind-address** *bind-address*

12. (Optional) Configure the source UDP port or a range of source UDP ports used for communication with the RADIUS server. If you do not specify a UDP port, the global UDP port is used.

    [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
    user@host# **set udp-port** *udp-port*

13. Configure the name of the RADIUS server to which the SAE sends packets for this plug-in.

    [edit shared sae group west-region configuration plug-ins name customRadiusAct custom-radius-accounting]
    user@host# **set default-peer** *default-peer*

14. (Optional) From operational mode, verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name
customRadiusAct custom-radius-accounting]
user@host# show
java-class-radius-packet-handler
net.juniper.smgt.radius.RadiusPacketHandlerImpl;
append-acct-status-type-attribute;
load-balancing-mode failover;
failback-timer -1;
timeout 15000;
retry-interval 3000;
maximum-queue-length 10000;
default-peer peer3;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*
- For information about setting up default peers, see *Creating RADIUS Peers* on page 138.

Configuring Authentication Plug-Ins

This section shows how to configure the authentication plug-ins described in Table 17. Because authentication and authorization are similar, the plug-in user interface does not distinguish between them. However, when you configure plug-ins, you need to set them up to perform the correct behavior, either authentication or authorization.

You can configure multiple authentication plug-ins. The plug-ins are called in an arbitrary order, and each plug-in can return authorization values. (If multiple plug-ins return a session-timeout value, the smallest value is used.) Authentication or authorization succeeds if all plug-in calls succeed.

Table 17: Authentication Plug-Ins

Plug-In	Description
Basic RADIUS authentication	Sends authentication information to an external RADIUS authentication server or a group of redundant servers. Java class name—net.juniper.smgt.sae.plugin.RadiusAuthPluginEventListener
Custom RADIUS authentication	Provides customized functions that can also be found in the flexible RADIUS authentication plug-ins. Custom plug-ins are internal plug-ins that are designed to deliver better system performance than the flexible RADIUS plug-ins. You can extend this plug-in by using the RADIUS client library. Java class name—net.juniper.smgt.sae.plugin.CustomRadiusAuth
Flexible RADIUS authentication	Performs the same functions as the basic RADIUS authentication plug-in, but also lets you customize RADIUS authentication packets that the SAE sends to RADIUS servers. You can specify which fields are included in RADIUS authentication packets and what information is contained in the fields. Java class name—net.juniper.smgt.sae.plugin.FlexibleRadiusAuthPluginEventListener

Table 17: Authentication Plug-Ins (continued)

Plug-In	Description
LDAP authentication	Performs authentication against different directories using different authentication methods. There are two LDAP authentication plug-ins: one authenticates subscribers, and the second authenticates SRC administrators so that they can access the SAE Web Admin application. Java class name of the subscriber authentication plug-in—net.juniper.smgt.sae.plugin.LdapAuthenticator Java class name of the administrator authentication plug-in—net.juniper.smgt.sae.plugin.adminLdap
Limiting subscribers	Limits the number of authenticated subscribers who connect to an IP interface on the router. Java class name—net.juniper.smgt.sae.plugin.LimitNumSubscriberPerIntfAuthPluginListener

Limiting Subscribers on Router Interfaces

You can limit the number of authenticated subscribers who connect to an IP interface on the router. This plug-in does not limit the number of unauthenticated subscribers who connect to an IP interface, and does not limit the number of subscribers who connect to a physical or link-layer interface. In the case of subscriber interfaces, the plug-in limits the number of authenticated subscribers on the subscriber interface but not on the underlying primary IP interface.

Use the following configuration statement to set up a plug-in that limits the number of subscribers who connect to interfaces:

```
shared sae configuration plug-ins name name interface-subscriber-limit {
    concurrent-subscribers concurrent-subscribers;
}
```

To set up a plug-in that limits the number of subscribers on interfaces:

1. From configuration mode, access the custom RADIUS accounting plug-in configuration. In this sample procedure, the plug-in called subsLimit is configured in the west-region SAE group.

   ```
   user@host# edit shared sae group west-region configuration plug-ins name
   subsLimit interface-subscriber-limit
   ```

2. Configure the number of authenticated subscribers who can connect to an IP interface on the router simultaneously.

   ```
   [edit shared sae group west-region configuration plug-ins name subsLimit
   interface-subscriber-limit]
   user@host# set concurrent-subscribers concurrent-subscribers
   ```

3. (Optional) Verify your configuration.

   ```
   [edit shared sae group west-region configuration plug-ins name subsLimit
   interface-subscriber-limit]
   user@host# show
   concurrent-subscribers 1;
   ```

Configuring Basic RADIUS Authentication Plug-Ins

You can use basic RADIUS authentication plug-ins to send authentication information to an external RADIUS accounting server or a group of redundant servers. To communicate with nonredundant servers, you need to create additional instances of the plug-in.

Use the following configuration statements to set up basic RADIUS authentication plug-ins:

```
shared sae configuration plug-ins name name radius-authentication {
    load-balancing-mode (failover | roundRobin);
    failback-timer failback-timer;
    nas-ip (SspIp | ErxIp);
    retry-interval retry-interval;
    maximum-queue-length maximum-queue-length;
    bind-address bind-address;
    udp-port udp-port;
    default-peer default-peer;
}
```

To set up basic RADIUS authentication plug-ins:

1. From configuration mode, access the basic RADIUS authentication plug-in configuration. In this sample procedure, the plug-in called RadiusAuth is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication**

2. Configure the mode for load-balancing RADIUS servers.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set load-balancing-mode** (failover | roundRobin)

3. Specify if and when the SAE attempts to fail back to the default peer.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set failback-timer** *failback-timer*

4. (Optional) Configure the value of the NAS-Ip attribute.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set nas-ip** (SspIp | ErxIp)

5. Configure the time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set retry-interval** *retry-interval*

6. Configure the maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set maximum-queue-length** *maximum-queue-length*

7. (Optional) Configure the source IP address that the plug-in uses to communicate with the RADIUS server. If you do not specify an address, the global default address is used.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set bind-address** *bind-address*

8. (Optional) Configure the source UDP port or a range of source UDP ports used for communication with the RADIUS server. If you do not specify a UDP port, the global UDP port is used.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set udp-port** *udp-port*

9. Configure the name of the RADIUS server to which the SAE sends packets for this plug-in.

   [edit shared sae group west-region configuration plug-ins name RadiusAuth radius-authentication]
   user@host# **set default-peer** *default-peer*

10. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name RadiusAuth
radius-authentication]
user@host# show
load-balancing-mode failover;
failback-timer -1;
retry-interval 3000;
maximum-queue-length 10000;
default-peer peer1;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*
- For information about setting up default peers, see *Creating RADIUS Peers* on page 138.

Configuring Flexible RADIUS Authentication Plug-Ins

Flexible RADIUS authentication plug-ins provide the same features as basic RADIUS authentication plug-ins. In addition, they allow you to customize RADIUS authentication packets that the system sends to RADIUS servers and specify which fields are included in the RADIUS authentication packets and what information is contained in the fields.

Use the following configuration statements to set up flexible RADIUS authentication plug-ins:

```
shared sae configuration plug-ins name name flex-radius-authentication {
    load-balancing-mode (failover | roundRobin);
    failback-timer failback-timer;
    timeout timeout;
    retry-interval retry-interval;
    maximum-queue-length maximum-queue-length;
    bind-address bind-address;
    udp-port udp-port;
    error-handling (0 | 1);
    default-peer default-peer;
    template template;
}
```

To set up flexible RADIUS authentication plug-ins:

1. From configuration mode, access the flexible RADIUS authentication plug-in configuration. In this sample procedure, the plug-in called flexRadiusAuth is configured in the west-region SAE group.

   ```
   user@host# edit shared sae group west-region configuration plug-ins name
   flexRadiusAuth flex-radius-authentication
   ```

2. Configure the mode for load-balancing RADIUS servers.

   ```
   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth
   flex-radius-authentication]
   user@host# set load-balancing-mode (failover | roundRobin)
   ```

3. Specify if and when the SAE attempts to fail back to the default peer.

   ```
   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth
   flex-radius-authentication]
   user@host# set failback-timer failback-timer
   ```

4. (Optional) Configure the maximum time the SAE waits for a response from a RADIUS server.

   ```
   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth
   flex-radius-authentication]
   user@host# set timeout timeout
   ```

5. Configure the time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth flex-radius-authentication]
   user@host# **set retry-interval** *retry-interval*

6. Configure the maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth flex-radius-authentication]
   user@host# **set maximum-queue-length** *maximum-queue-length*

7. (Optional) Configure the source IP address that the plug-in uses to communicate with the RADIUS server. If you do not specify an address, the global default address is used.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth flex-radius-authentication]
   user@host# **set bind-address** *bind-address*

8. (Optional) Configure the source UDP port or a range of source UDP ports used for communication with the RADIUS server. If you do not specify a UDP port, the global UDP port is used.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth flex-radius-authentication]
   user@host# **set udp-port** *udp-port*

9. Configure the way the SAE handles errors.

   [edit shared sae group west-region configuration plug-ins name flexRadiusAuth flex-radius-authentication]
   user@host# **set error-handling** (0 | 1)

10. Configure the name of the RADIUS server to which the SAE sends packets for this plug-in.

    [edit shared sae group west-region configuration plug-ins name flexRadiusAuth flex-radius-authentication]
    user@host# **set default-peer** *default-peer*

11. Configure the name of the RADIUS packet template that defines attributes for this plug-in.

    [edit shared sae group west-region configuration plug-ins name flexRadiusAct flex-radius-accounting]
    user@host# **set template** *template*

12. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name
flexRadiusAuth flex-radius-authentication]
user@host# show
load-balancing-mode failover;
failback-timer -1;
timeout 15000;
retry-interval 3000;
maximum-queue-length 10000;
error-handling 0;
default-peer 1;
template stdAuth;
peer-group 1 {
  server-address ;
  server-port 1812;
  secret ********;
}
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*
- For information about setting up default peers, see *Creating RADIUS Peers* on page 138.
- For information about defining RADIUS packet templates, see *Defining RADIUS Packets for Flexible RADIUS Plug-Ins with the SRC CLI* on page 163.

Configuring Custom RADIUS Authentication Plug-Ins

The custom RADIUS authentication plug-ins provide the same functions as the flexible RADIUS authentication plug-ins, but are designed to deliver better system performance. To use a custom plug-in, you must provide a Java class that implements the SPI defined in the RADIUS client library. Use this SPI to specify which fields and field values to include in RADIUS accounting packets. The RADIUS client library is part of the SAE core API.

See the documentation for the RADIUS client library in the SRC software distribution in the folder *SDK/doc/sae/net/juniper/smgt/sae/radiuslib* or the SAE core API documentation on the Juniper Networks Web site at

http://www.juniper.net/techpubs/software/management/sdx/api-index.html

For a sample implementation, see the following directory in the SRC software distribution:
SDK/plugin/java/src/net/juniper/smgt/sample/radiuslib/RadiusPacketHandlerImpl.java.

Use the following configuration statements to set up custom RADIUS authentication plug-ins:

```
shared sae configuration plug-ins name name custom-radius-authentication {
    java-class-radius-packet-handler java-class-radius-packet-handler;
    class-path-radius-packet-handler class-path-radius-packet-handler;
    require-mandatory-attributes;
    load-balancing-mode (failover | roundRobin);
    failback-timer failback-timer;
    timeout timeout;
    retry-interval retry-interval;
    maximum-queue-length maximum-queue-length;
    bind-address bind-address;
    udp-port udp-port;
    default-peer default-peer;
}
```

To set up custom RADIUS authentication plug-ins:

1. From configuration mode, access the custom RADIUS authentication plug-in configuration. In this sample procedure, the plug-in called customRadiusAuth is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication**

2. Configure the name of the Java class that implements the RadiusPacketHandler interface in the RADIUS client library.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set java-class-radius-packet-handler** *java-class-radius-packet-handler*

3. Configure the URLs that identify a location from which Java classes are loaded when the plug-in is initialized.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set class-path-radius-packet-handler** *class-path-radius-packet-handler*

4. (Optional) Specify that a RADIUS authentication or accounting request must contain all mandatory RADIUS attributes before sending the request packet.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set require-mandatory-attributes**

5. Configure the mode for load-balancing RADIUS servers.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set load-balancing-mode** (failover | roundRobin)

6. Specify if and when the SAE attempts to fail back to the default peer.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set failback-timer** *failback-timer*

7. (Optional) Configure the maximum time the SAE waits for a response from a RADIUS server.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set timeout** *timeout*

8. Configure the time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set retry-interval** *retry-interval*

9. Configure the maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.

   [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
   user@host# **set maximum-queue-length** *maximum-queue-length*

10. (Optional) Configure the source IP address that the plug-in uses to communicate with the RADIUS server. If you do not specify an address, the global default address is used.

    [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
    user@host# **set bind-address** *bind-address*

11. (Optional) Configure the source UDP port or a range of source UDP ports used for communication with the RADIUS server. If you do not specify a UDP port, the global UDP port is used.

    [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
    user@host# **set udp-port** *udp-port*

12. Configure the name of the RADIUS server to which the SAE sends packets for this plug-in.

    [edit shared sae group west-region configuration plug-ins name customRadiusAuth custom-radius-authentication]
    user@host# **set default-peer** *default-peer*

13. (Optional) From operational mode, verify your configuration.

```
[edit shared sae configuration plug-ins name customRadiusAuth
custom-radius-authorization]
user@host# show
java-class-radius-packet-handler
net.juniper.smgt.radius.RadiusPacketHandlerImpl;
require-mandatory-attributes;
load-balancing-mode failover;
failback-timer -1;
timeout 15000;
retry-interval 3000;
maximum-queue-length 10000;
default-peer peer4;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*
- For information about setting up default peers, see *Creating RADIUS Peers* on page 138.

Configuring LDAP Authentication Plug-Ins

Use the following configuration statements to configure LDAP authentication plug-ins:

```
shared sae configuration plug-ins name name ldap-authentication {
    method (search | bind);
    server server;
    bind-dn bind-dn;
    bind-password bind-password;
    search-filter search-filter;
    (ldaps);
    search-base-dn search-base-dn;
    name-attribute name-attribute;
    password-attribute password-attribute;
    service-bundle-attribute service-bundle-attribute;
    session-volume-quota session-volume-quota;
    timeout timeout;
}
```

To create LDAP authentication plug-ins:

1. From configuration mode, access the custom LDAP authentication plug-in configuration. In this sample procedure, the plug-in called ldapAuth is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication**

2. Configure the LDAP authentication method that the SAE uses.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set method** (search | bind)

3. (Optional) Configure a comma-separated list of IP addresses or hostnames of the LDAP authentication server.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set server** *server*

4. (Optional) Configure the DN used to authenticate access to the directory.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set bind-dn** *bind-dn*

5. (Optional) Configure the password that the SAE uses to authenticate its access to the directory to search for the subscriber profile. If you do not specify a bind DN or bind password, the SAE uses anonymous access.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set bind-password** *bind-password*

6. (Optional) Configure the additional LDAP search filter that the SAE uses to search the directory for the subscriber profile.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set search-filter** *search-filter*

7. (Optional) Enable the secure protocol used for LDAP connections with the directory. LDAPS, the only secure protocol supported, causes communication with the directory to be encrypted with Secure Sockets Layer (SSL).

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set ldaps**

8. (Optional) Configure the base DN for searching entries in the directory.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set search-base-dn** *search-base-dn*

9. (Optional) Configure the name of the directory attribute that holds the username.

   [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
   user@host# **set name-attribute** *name-attribute*

10. (Optional) Configure the name of the directory attribute that stores the password.

    [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
    user@host# **set password-attribute** *password-attribute*

11. (Optional) Configure the name of the directory attribute that contains the name of the service bundle that is used for subscriber authentication. This value is made available to the subscriber classification process and can be used to select the subscriber profile to load.

    [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
    user@host# **set service-bundle-attribute** *service-bundle-attribute*

12. (Optional) Configure the name of the LDAP attribute that contains the value of the session volume quota. The LDAP plug-in sets the session volume quota to this value.

    [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
    user@host# **set session-volume-quota** *session-volume-quota*

13. (Optional) Configure the maximum time the SAE waits for a response from a directory server.

    [edit shared sae group west-region configuration plug-ins name ldapAuth ldap-authentication]
    user@host# **set timeout** *timeout*

14. (Optional) From operational mode, verify your configuration.

```
[edit shared sae group west-region configuration plug-ins name ldapAuth
ldap-authentication]
user@host# show
method search;
search-filter (objectClass=umcSubscriber);
name-attribute uniqueId;
timeout 5000;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*

Configuring UDP Ports for RADIUS Plug-Ins

In RADIUS packets that RADIUS plug-ins send to a RADIUS server, the plug-in uses an identifier field to match requests to replies. This field provides for a maximum of 256 identifiers. Once all identifiers are used, the plug-in cannot send any more requests until it receives replies that match the requests already sent. In high-load systems, this limit can slow performance.

To overcome this limitation, you can configure a pool of UDP ports for RADIUS plug-ins. Having a pool of ports allows RADIUS plug-ins to create one queue per port to wait for RADIUS replies. Each queue can wait for 256 RADIUS packets. The RADIUS plug-ins send RADIUS packets through the pool of ports in a round-robin mode.

You can configure a global source UDP port or pool of ports that RADIUS plug-ins use to communicate with RADIUS servers. You can also configure UDP ports for each plug-in instance. If you do not configure a UDP port for a plug-in instance, the plug-in uses the global UDP port.

Configuring Global UDP Ports

Use the following configuration statement to configure global configuration ports:

```
shared sae configuration global-radius-udp-port {
    udp-port;
}
```

To configure global UDP ports:

1. From configuration mode, access the global RADIUS UDP port configuration. In this sample procedure, the UDP port is configured in the west-region SAE group.

   ```
   user@host# edit shared sae group west-region configuration
   global-radius-udp-port
   ```

2. Configure the source UDP port or a pool of ports that RADIUS plug-ins use to communicate with RADIUS servers.

   ```
   [edit shared sae group west-region configuration global-radius-udp-port]
   user@host# set udp-port
   ```

Defining RADIUS Packets for Flexible RADIUS Plug-Ins with the SRC CLI

Flexible RADIUS accounting and authentication plug-ins allow you to define the content of RADIUS packets that the SAE sends to RADIUS servers. You can specify which attributes are included in different types of RADIUS packets (for example, session start or stop requests, or accounting on or off requests). You can also specify what information is contained in the attribute fields.

A RADIUS attribute configuration consists of RADIUS attribute instances. Each instance defines attributes for a specific type of packet—For example, start requests or accounting off requests.

Within each attribute instance, you define individual RADIUS attributes. The following is a RADIUS attribute instance for authentication requests:

```
radius-attributes auth {
  attributes {
    User-Name loginId;
    User-Password password;
    NAS-Identifier localNasId;
    NAS-IP-Address localNasIp;
    NAS-Port nasPort;
  }
}
```

Each RADIUS packet template can consist of multiple RADIUS attribute instances.

Using Default RADIUS Templates

The SRC software comes with two default templates:

- stdAcct—Defines RADIUS accounting packets and is used in the default RADIUS flexible accounting plug-in instance flexRadiusAcct.
- stdAuth—Defines RADIUS authentication packets and is used in the default RADIUS flexible authentication plug-in instance flexRadiusAuth.

Naming RADIUS Attribute Instances

Attribute instances define attributes for a specific type of RADIUS packet. The name that you assign to an attribute instance specifies the type of packet to which the attribute definition is applied. Table 18 lists the available packet types.

Table 18: RADIUS Attribute Instance Names

Attribute Instance (Packet-Type)	Type of RADIUS Packet to Which Attribute Definition Is Applied
acct	Any accounting request
auth	Any authentication request
authresp	Any authorization response
dhcpresp	DHCP response
off	Accounting-Off requests
on	Accounting-On requests
onoff	Accounting-On or Accounting-Off requests
start	Start requests
startstop	Start, Stop, or Interim Update requests
stop	Stop or Interim Update requests
svcacct	Service Session Start, Stop, or Interim requests
svcresp	Any service authorization response
svcstart	Service Session Start requests
svcstop	Service Session Stop or Interim requests
useracct	Subscriber Session Start, Stop, or Interim requests
userresp	Any subscriber authorization response
userstart	Subscriber Session Start requests
userstop	Subscriber Session Stop, or Interim requests

Defining RADIUS Attributes

RADIUS attribute definitions consist of a RADIUS attribute and a value for the RADIUS attribute.

You can define values for standard RADIUS attributes or JUNOSe vendor-specific attributes (VSAs).

Standard RADIUS Attributes

For standard RADIUS attributes, use a name or number as defined in RFC 2865—Remote Authentication Dial In User Service (RADIUS) (June 2000), RFC 2866—RADIUS Accounting (June 2000), or RFC 2869—RADIUS Extensions (June 2000). For a full list, see www.iana.org/assignments/radius-types.

Juniper Networks VSAs

For Juniper Networks VSAs, use one of the following formats:

- Vendor-Specific.4874.<vsa#>.<type>
- 26.4874.<vsa#>.<type>

where <type> is one of the following:

- text—Indicates that the value is 1–253 octets containing UTF-8 encoded characters
- string—Indicates that the value is 1–253 octets containing binary data
- address—Indicates that the value is a 32-bit value
- integer—Indicates that the value is a 32-bit unsigned value
- time—Indicates that the value is a 32-bit unsigned value, seconds since 00:00:00 UTC, January 1, 1970

The following is an example of RADIUS attribute instances that define RADIUS VSAs.

```
radius-attributes svcresp {
  attributes {
    Session-Timeout setSessionTimeout(ATTR);
    Idle-Timeout setIdleTimeout(ATTR);
    vendor-specific.Juniper.Sdx-Session-Volume-Quota setSessionVolumeQuota(ATTR);
    vendor-specific.WISPr.Redirection-URL "setProperty(\"startURL=%s\" % ATTR)";
    vendor-specific.WISPr.Bandwidth-Min-Up "setSubstitution(\"min_up_rate=%s\" % ATTR)";
    vendor-specific.WISPr.Bandwidth-Min-Down "setSubstitution(\"min_down_rate=%s\" % ATTR)";
    vendor-specific.WISPr.Bandwidth-Max-Up "setSubstitution(\"max_up_rate=%s\" % ATTR)";
    vendor-specific.WISPr.Bandwidth-Max-Down "setSubstitution(\"max_down_rate=%s\" % ATTR)";
  }
}

radius-attributes dhcpresp {
  attributes {
    Framed-Pool setPoolName(ATTR);
    Framed-IP-Address setUserIpAddress(ATTR);
    26.4874.1.text setAuthVirtualRouterName(ATTR);
    26.4874.2.text setPoolName(ATTR);
    26.4874.31.text setServiceBundle(ATTR);
  }
}
```

Defining the Values of RADIUS Attributes

The values of RADIUS attributes can be a standard value (see Table 19) or an expression. Expressions are evaluated with Python. For example: lowWord(inOctets) extracts the lower 32 bits of the 64-bit inOctets counter. You can define multiple values for an expression in a comma-separated list.

Table 19: Standard Values for RADIUS Attributes

Value	Type of Plug-In	Comments
accountingId	User and service tracking	
authUserId	Service tracking	
dhcp	User and service tracking	Provides access to DHCP packet. See Table 12 on page 73 for details.
domain	Authorization	
eventTime	User and service tracking	Seconds since 1970-01-01T00:00Z
ifRadiusClass	User and service tracking	
ifSessionId	User and service tracking	
inOctets	Service tracking	64-bit counter
inPackets	Service tracking	
interfaceAlias	User and service tracking	
interfaceDescr	User and service tracking	
interfaceName	User and service tracking	
localNasId	All	Configured NAS-ID
localNasIp	All	Configured NAS-IP
loginId	User and service authorization	ID provided by the subscriber; the loginId value is not separated into UID and domain name.
loginName	User and service tracking	Name that the subscriber uses to log in to portal
nasIp	User and service tracking	NAS IP address of the router
nasPort	User and service tracking	32-bit integer
outOctets	Service tracking	64-bit counter
outPackets	Service tracking	
password	User and service authorization	
portId	User and service tracking	ID of the port on the JUNOSe router; for example, FastEthernet 3/1:2001
primaryUserName	User and service tracking	Name that the subscriber uses for DHCP/PPP authentication
radiusClass	User tracking, user and service authorization	For service tracking, this value is taken from the RADIUS Access-Accept response. If the response does not contain a value, the RADIUS class defined in the service definition is used. This attribute can be set by an authorization response.
replyMessage	User and service authorization	This attribute can only be set.
routerName	User and service tracking	
serviceBundle	User tracking and authorization	This attribute can be set by an authorization response.

Table 19: Standard Values for RADIUS Attributes (continued)

Value	Type of Plug-In	Comments
serviceName	Service tracking	Sets an arbitrary attribute (for example, class) to the name of the service.
serviceSessionName	Service tracking	Named service session; empty for default session
serviceSessionTag	Service tracking	
sessionId	User and service tracking	
sessionTime	User and service tracking	
sessionTimeout	User tracking, user and service authorization	This attribute can be set by an authorization response.
sessionVolumeQuota	User authorization	This attribute can only be set. It is sent for session tracking events and can be returned by service authorization events. It can be set and retrieved through the portal API and can also be defined through an LDAP attribute in the service definition. If the attribute is defined multiple times, the following precedence is observed: 1. Service definition (lowest) 2. Authorization 3. API call (highest) **NOTE:** The SAE does not enforce a volume quota directly; it only makes the attribute available to an external application that can control the volume quota.
setAcctInterimTime	User authorization	Integer
setAuthVirtualRouterName	DHCP authorization	Text
setIdleTimeout(ATTR)	User authorization	
setLoadServices(ATTR)	User authorization	This attribute can only be set.
setPoolName	DHCP authorization	Text
setRadiusClass(ATTR)	User and service authorization	
setReplyMessage(ATTR)	User and service authorization	
setSessionTimeout(ATTR)	User and service authorization	
setServiceBundle(ATTR)	User authorization	
setSessionVolumeQuota(ATTR)	User authorization	
setSubstitution	User authorization	Text. Substitutions can be set only for service sessions.
setTerminateTime	User authorization	Text
setUserIpAddress	DHCP authorization	Integer
sspHost	User and service tracking	
terminateCause	User and service tracking	
uid	User and service authorization	
userDn	User and service tracking	
userIpAddress	User and service tracking	
userMacAddress	User and service tracking	
userRadiusClass	Service tracking	RADIUS class of associated subscriber session
userSessionId	Service tracking	RADIUS session ID of associated subscriber session

Configuring a RADIUS Packet Template

There are two ways to define RADIUS packets for flexible RADIUS accounting and authentication plug-ins:

- Define attributes in a template, and then apply the template to flexible RADIUS accounting and authentication plug-ins.
- Define attributes in the packet definition configuration of a flexible plug-in instance. These definitions override definitions in packet templates.

Use the following configuration statements to configure a RADIUS packet template:

shared sae configuration radius-packet-template *name* ...

shared sae configuration radius-packet-template *name* radius-attributes *name* ...

shared sae configuration radius-packet-template *name* radius-attributes *name*
attributes *name* {
    value;
}

shared sae configuration plug-ins name *name* flex-radius-accounting
radius-packet-definition *name* ...

shared sae configuration plug-ins name *name* flex-radius-accounting
radius-packet-definition *name* attributes *name* {
    value;
}

shared sae configuration plug-ins name *name* flex-radius-authentication
radius-packet-definition *name* ...

shared sae configuration plug-ins name *name* flex-radius-authentication
radius-packet-definition *name* attributes *name* {
    value;
}

To configure a template:

1. From configuration mode, access the RADIUS packet template configuration. In this sample procedure, the stdAcct template is configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration radius-packet-template stdAcct**

2. Create an attribute instance using the names in Table 18 on page 164, and enter the configuration for the RADIUS attribute instance.

   [edit shared sae group west-region configuration radius-packet-template stdAcct]
   user@host# **edit radius-attributes** *name*

3. Add RADIUS attribute definitions to the attribute instance. Repeat this step for each attribute.

[edit shared sae group west-region configuration radius-packet-template stdAcct radius-attributes svcstop]
user@host# **set attributes** *name value*

For example:

[edit shared sae group west-region configuration radius-packet-template stdAcct radius-attributes svcstop]
user@host# **set attributes Acct-Session-ID sessionId**

4. (Optional) Verify the configuration of your attribute instance.

```
[edit shared sae group west-region configuration radius-packet-template
stdAcct radius-attributes svcstop]
user@host# show
attributes {
  Acct-Input-Octets lowWord(inOctets);
  Acct-Output-Octets lowWord(outOctets);
  Acct-Input-Packets lowWord(inPackets);
  Acct-Output-Packets lowWord(outPackets);
  Acct-Input-Gigawords highWord(inOctets);
  Acct-Output-Gigawords highWord(outOctets);
}
```

5. (Optional) Verify the configuration of the RADIUS packet template.

```
[edit shared sae group west-region configuration radius-packet-template
stdAcct radius-attributes svcstop]
user@host# up
[edit shared sae group west-region configuration radius-packet-template
stdAcct]
user@host# show
radius-attributes svcstop {
  attributes {
    Acct-Input-Octets lowWord(inOctets);
    Acct-Output-Octets lowWord(outOctets);
    Acct-Input-Packets lowWord(inPackets);
    Acct-Output-Packets lowWord(outPackets);
    Acct-Input-Gigawords highWord(inOctets);
    Acct-Output-Gigawords highWord(outOctets);
  }
}
radius-attributes stop {
  attributes {
    Acct-Session-Time sessionTime;
    Acct-Terminate-Cause terminateCause;
  }
}
radius-attributes svcacct {
  attributes {
    Class radiusClass;
  }
}
radius-attributes acct {
  attributes {
    Acct-Session-Id sessionId;
    NAS-Identifier localNasId;
    NAS-IP-Address localNasIp;
    Event-Time eventTime;
  }
}
```

```
radius-attributes startstop {
  attributes {
    Acct-Multi-Session-Id ifSessionId;
    NAS-Port-Id "\"%s %s\" %(routerName, portId or interfaceName)";
    NAS-Port "nasPort or None";
  }
}
```

More About Using Flexible RADIUS Packet Definitions

This section shows some of the ways you can use flexible RADIUS packet definitions. Remember that the name of the attribute instance determines the type of RADIUS packet in which the packet definition is used.

- To use the Challenge Handshake Authentication Protocol (CHAP) to authenticate subscribers, include the Chap-Password and optionally the Chap-Challenge attributes in authentication requests. (We recommend that you use Chap-Password only. Use Chap-Challenge only if required.) To use a CHAP password, include the following in attribute instance auth:

  Chap-Password = password

- To cause the Calling-Station-Id attribute to use the subscriber's MAC address:

  Calling-Station-Id = userMacAddress

- To set the value to prefix N followed by the service name and the prefix S followed by the service session name:

  'N'+serviceName, 'S'+serviceSessionName

- To construct a value for the Nas-Port-Id attribute by concatenating the value of routerName, a space, and the Nas-Port-ID on the router:

  Nas-Port-Id=routerName + " " + portId

  For example, the constructed value might be:

  default@phoenix FastEthernet 4/2

  - The following example sets the User-Name attribute as follows:
  - Sets the value to accountingId, or
  - If accountingId is empty, sets the value to loginName, or
  - If loginName is also empty, sets the value to NN

  User-Name = accountingId or loginName or "NN"

  - To extract the lower 32 bits of the 64-bit inOctet counter:

  Acct-Input-Octets = lowWord(inOctets)

- To set the counter fields in the RADIUS packet to the appropriate 32-bit values:

  ```
  Acct-Input-Octets = lowWord(inOctets)
  Acct-Output-Octets = lowWord(outOctets)
  Acct-Input-Packets = inPackets
  Acct-Output-Packets = outPackets

  Acct-Input-Gigawords = highWord(inOctets)
  Acct-Output-Gigawords = highWord(outOctets)
  ```

  - The inOctets and outOctets are 64-bit values and must be split into lower 32-bit (Acct-*-Octets) and upper 32-bit (Acct-*-Gigawords) values.
  - The inPacket and outPacket counters are 32-bit values and can be assigned directly.

Setting Values in Authentication Response Packets

You can use some special attribute values to set values in authentication response packets. For example:

- setRadiusClass(ATTR)
- setSessionTimeout(ATTR)
- setSessionVolumeQuota(ATTR)

Table 19 on page 166 lists the type of packets (authresp, userresp, or svcresp) in which you can use these values.

When the RADIUS client finds one of these attribute values in an authentication response, it binds ATTR to the current attribute and executes the defined expression. The expression calls one of the available set methods to set the value in the plug-in event.

Below are some examples.

- To set a session timeout:

  ```
  Session-Timeout = setSessionTimeout(ATTR)
  ```

- To set the RADIUS class:

  ```
  Class = setRadiusClass(ATTR)
  ```

- To set the service bundle in VSA 31:

  ```
  26.4874.31.text = setServiceBundle(ATTR)
  ```

- To set the session volume quota:

  ```
  26.4874.50.text = setSessionVolumeQuota(ATTR)
  ```

Selecting IP Address Pools Using DHCP Response Packets

For DHCP subscribers, you can set up RADIUS authorization plug-ins to return to the router attributes that can be used to select a DHCP address such as framed IP address and pool. You can also set up the name of the virtual router on which the address pool is located and select a fixed address for each subscriber.

- Framed IP address—Selects the pool from which the address is allocated; if the framed IP address is not available, the DHCP server allocates the next available address in the pool; use the setUserIpAddress value.
- Framed IP pool—Name of the address pool on the router from which an IP address is assigned; use the setPoolName value.
- Virtual router name—Name of the virtual router on which the address pool is located; use the setAuthVirtualRouterName value.

You can also select a fixed address for each subscriber. If you identify subscribers by port information (for example, NAS-IP and NAS-Port), the authorization response can select a fixed IP address for each subscriber.

NOTE: Parameters set in the DHCP profile override parameters set by DHCP authorization plug-ins.

Configuring Event Publishers

This topic shows how to configure event publishers. It covers the following tasks:

Configuring Global and Default Retailer Event Publishers

Use the following configuration statements to configure global and default retailer event publishers.

```
shared sae configuration plug-ins event-publishers {
    subscriber-authorization subscriber-authorization;
    default-retailer-authentication default-retailer-authentication;
    default-retailer-dhcp-authentication default-retailer-dhcp-authentication;
    dhcp-authorization dhcp-authorization;
    service-authorization service-authorization;
    subscription-authorization subscription-authorization;
    subscriber-tracking subscriber-tracking;
    service-tracking service-tracking;
    interface-tracking interface-tracking;
    embedded-admin-server-authorization embedded-admin-server-authorization;
}
```

To configure global and default retailer event publishers:

1. From configuration mode, access the event publisher configuration. In this sample procedure, the event publishers are configured in the west-region SAE group.

   user@host# **edit shared sae group west-region configuration plug-ins event-publishers**

2. Configure plug-ins that authorize subscriber sessions.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set subscriber-authorization** *subscriber-authorization*

3. Configure plug-ins that authenticate subscribers who are assigned to retailer objects that do not specify an authentication plug-in.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set default-retailer-authentication** *default-retailer-authentication*

4. Configure plug-ins that authenticate DHCP address requests for subscribers who are assigned to retailer objects that do not specify a DHCP authorization plug-in.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set default-retailer-dhcp-authentication**
   default-retailer-dhcp-authentication

5. Configure plug-ins that authorize all DHCP address requests for all DHCP subscribers who log in to a portal.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set dhcp-authorization** *dhcp-authorization*

6. Configure plug-ins that authorize all service sessions.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set service-authorization** *service-authorization*

7. Configure plug-ins that authorize subscribers to change their subscriptions.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set subscription-authorization** *subscription-authorization*

8. Configure plug-ins that collect accounting data for all subscriber sessions.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set subscriber-tracking** *subscriber-tracking*

9. Configure plug-ins that collect accounting data for all service sessions.

   [edit shared sae group west-region configuration plug-ins event-publishers]
   user@host# **set service-tracking** *service-tracking*

10. Configure plug-ins, including network information collector (NIC) SAE plug-in agents, that collect accounting data for all interfaces that the SAE manages.

    [edit shared sae group west-region configuration plug-ins event-publishers]
    user@host# **set interface-tracking** *interface-tracking*

11. Configure plug-ins that authorize administrators to connect to the embedded Web server, which is used to access SAE Web Admin.

    [edit shared sae group west-region configuration plug-ins event-publishers]
    user@host# **set embedded-admin-server-authorization** *embedded-admin-server-authorization*

12. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration plug-ins event-publishers]
user@host# show
subscriber-authorization ;
default-retailer-authentication ldapAuth;
default-retailer-dhcp-authentication ;
dhcp-authorization ;
service-authorization ;
subscription-authorization ;
subscriber-tracking fileAcct;
service-tracking fileAcct;
interface-tracking ;
embedded-admin-server-authorization adminLdap;
```

Related Topics

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 21, Setting Up an SAE with the SRC CLI.*

Configuring Service-Specific Event Publishers

In the value-added services definition, you can configure two event publishers for a service:

- Authorization plug-ins—Authenticate subscribers of the service and/or authorize service sessions for this service. These plug-in instances are called before a subscription to this service is activated.

- Tracking plug-ins—Track service sessions of this service. These plug-in instances are called when a service session is started and stopped and during interim updates.

See *SRC-PE Services and Policies Guide, Chapter 1, Managing Services with the SRC CLI*.

Configuring Retailer-Specific Event Publishers

In the retailer definition, you can configure three event publishers for a retailer:

- Authentication plug-ins—Authenticate subscribers who log in to the domains of the retailer. These plug-in instances are called when a subscriber tries to log in to the SAE through the portal login.

  If you do not specify retailer-specific authentication plug-ins, the default retailer authentication plug-ins are called. If you do not specify default retailer authentication plug-ins, subscribers are admitted without authentication.

- Tracking plug-ins—Track sessions of subscribers who log in to the domains of the retailer. These plug-in instances are called after a subscriber session has started and when the session is stopped.

- DHCP authorization plug-ins—Authenticate DHCP address requests for subscribers who log in to the domains of the retailer.

See *Adding Retailers* on page 208.

Configuring Virtual Router–Specific Event Publishers

In the virtual router definition, you can configure an interface-tracking plug-in event publisher for a virtual router. These plug-in instances are called when a managed interface is started and stopped. They are called after an interface comes up, when new policies are installed on the interface, and when the interface goes down.

For information about configuring virtual routers for JUNOSe routers, see *SRC-PE Network Guide, Chapter 5, Using JUNOSe Routers in the SRC Network with the SRC CLI*.

For information about configuring virtual routers for JUNOS routing platforms, see *SRC-PE Network Guide, Chapter 7, Using JUNOS Routing Platforms in the SRC Network with the SRC CLI*.