Chapter 11

Configuring Authorization and Accounting Plug-Ins with SDX Configuration Editor

This chapter describes how to configure accounting and authorization plug-ins with SDX Configuration Editor. It also describes how to configure global and default retailer event publishers.

You can also configure plug-ins with the SRC CLI. See *Chapter 12, Configuring Accounting and Authentication Plug-Ins with the SRC CLI*.

Topics in this chapter include:

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* in ***SRC-PE Solutions Guide, Chapter 6, Configuring the SAE for a PCMM Environment with SDX Configuration Editor***. 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***. Java class name—net.juniper.smgt.sae.plugin.qtp.QosProfileTrackingPluginEventListener

The overall steps to configure a tracking plug-in are:

1. Create and configure a plug-in instance in the plug-in pool. The following sections show how to create and configure an instance for each type of tracking plug-in.

2. Configure an event publisher to publish events to the plug-in instance.

   See *Configuring Event Publishers* on page 184.

Configuring Flat File Accounting Plug-Ins

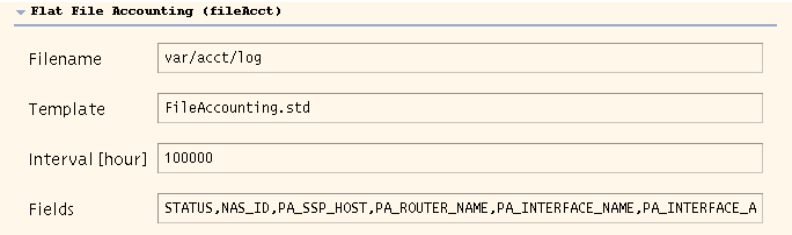
Flat file accounting plug-ins write information to a file in a comma-separated format. The SRC software has a default flat file accounting plug-in instance called fileAcct. The fileAcct instance logs all possible attributes for 24-hour periods in the file *var/acct/log*. You can modify the fileAcct instance, use it as is, or create a new instance.

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 156.

To create flat-file accounting plug-in instances:

1. In the Plug-In Pool area of the Plug-Ins pane, create a flat file accounting instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the fields for the plug-in instance as described below.

Filename

- Name and location of the file to which the SAE writes accounting information. The SAE names accounting files by appending the timestamp for the start of the accounting period.
- Value—Path and name of file
- Default—*var/acct/log*
- Property name—File

Template

- Name of a template that defines header names for attributes listed in accounting files. See *Configuring Headers for Flat File Accounting Plug-Ins* on page 156.
- Value—Name of the template in the format FileAccounting.<template name>
- Default—FileAccounting.std
- Property name—Template

Interval [hour]

- Number of hours of information stored in each accounting file. When the interval expires, the SAE closes the file, renames it to the archive name, and creates a new file.
- Accounting files are aligned with midnight of the day the SAE process starts. If the interval is 24 hours, the SAE starts a new file at midnight every day beginning on the day that the SAE starts.
  - If the interval is a divisor of 24 hours (for example, 15 minutes, 30 minutes, 1 hour), there is a repeatable pattern of file starts. For example, if the interval is set to 6 hours, the SAE creates a new file at midnight, 6 am, 12 am, and 6 pm every day.
  - If the interval is not a divisor of 24, then the file-start times shift each day to different times of the day.
- If the SAE is restarted, the schedule for creating accounting files is reset to start at midnight.
- Value—Integer in the format < hour > [":" < minute >]; there are no restrictions on interval length, but we recommend that you set a value that is a divisor of 24 hours
- Default—24
- Property name—Interval

Fields

- Attributes to be recorded as fields in the accounting file.
- Value—Comma-separated list of any of the following attributes:
  - NAS_ID—Identifier of the SAE (configurable)
  - PA_ACCOUNTING_ID—Accounting ID attribute from LDAP
  - PA_AGGR_ACCOUNTING_ID—Accounting ID of the subscriber who started the aggregate service session
  - PA_AGGR_AUTH_USER_ID—Subscriber ID that was used to authenticate the aggregate service session
  - PA_AGGR_LOGIN_NAME—Login name of subscriber who started the aggregate service session
  - PA_AGGR_SESSION_ID—Accounting session ID of the aggregate service session
  - PA_AGGR_USER_DN—Subscriber profile DN of the subscriber who started the aggregate service session
  - PA_AGGR_USER_IP—IP address of the subscriber who started the aggregate service session
  - PA_AUTH_USER_ID—Subscriber ID used for service authentication
  - PA_DOMAIN—Domain for secondary authentication
  - PA_DOWNSTREAM_BANDWIDTH—Downstream bandwidth for the service
  - PA_EVENT_TIME—Timestamp when the event was created

- PA_EVENT_TIME_MILLISECOND—Number of milliseconds since midnight 1970-01-01 UTC
- PA_IF_INDEX—SNMP index of the router interface
- PA_IF_RADIUS_CLASS—RADIUS class of the router interface
- PA_IF_SESSION_ID—Session ID assigned by the router
- PA_IN_OCTETS—Number of octets received from the subscriber (64 bit)
- PA_IN_PACKETS—Number of packets received from the subscriber (64 bit)
- PA_INTERFACE_ALIAS—Alias of router interface
- PA_INTERFACE_DESCR—Description of router interface
- PA_INTERFACE_NAME—Name of router interface
- PA_LOGIN_ID—Subscriber's login ID
- PA_LOGIN_NAME—Name of logged-in subscriber
- PA_NAS_IP—IP address that the router uses for accounting
- PA_NAS_INET_ADDRESS—IP address of the router that uses a byte array instead of an integer
- PA_NAS_PORT—Identifier that the router uses to identify the interface to RADIUS
- PA_OPERATIONAL—Flag that identifies whether an interface was operational at the time of the tracking event
- PA_OUT_OCTETS—Number of octets sent to the subscriber (64 bit)
- PA_OUT_PACKETS—Number of packets sent to the subscriber (64 bit)
- PA_PASSWORD—Password for secondary authentication
- PA_PORT_ID—Identifier of the physical interface (VirtualRouter@ERX interface slot/port.sub)
- PA_PRIMARY_USER_NAME—pppLoginName or public DhcpUserName
- PA_PROPERTY—Session property
- PA_RADIUS_CLASS—RADIUS class attribute
- PA_REPLY_MESSAGE—Message that a plug-in returns to the SAE during authorization
- PA_RETAILER_DN—Retailer DN associated with the domain
- PA_ROUTER_NAME—Name of the router
- PA_SERVICE_BUNDLE—RADIUS vendor-specific attribute (VSA) that a user authorization plug-in returns to the SAE
- PA_SERVICE_NAME—Name of SAE service
- PA_SERVICE_SCOPE—List of service scopes
- PA_SERVICE_SESSION_NAME—Name of dynamic service session
- PA_SERVICE_SESSION_TAG—Tag string assigned to dynamic service session
- PA_SESSION_ID—Session ID assigned by the SAE
- PA_SESSION_TIMEOUT—Number of seconds that the session is up

- PA_SESSION_VOLUME_QUOTA—Amount of data that a subscriber is allowed to upload or download
- PA_SSP_HOST—Hostname of the SAE server
- PA_SUBSCRIPTION_NAME—Name of the subscription
- PA_SUBSTITUTION—Parameter substitution set by a service or user authorization plug-in
- PA_TERMINATE_CAUSE—RADIUS termination cause (See RFC 2866—RADIUS Accounting (June 2000)—for possible values.)
- PA_TERMINATE_TIME—Time to end a subscriber session
- PA_UID—Subscriber ID used for secondary authentication
- PA_UPSTREAM_BANDWIDTH—Upstream bandwidth for the service
- PA_USER_DN—DN of the subscriber profile
- PA_USER_INET_ADDRESS—IP address of the subscriber that uses a byte array instead of an integer
- PA_USER_IP_ADDRESS—IP address of subscriber
- PA_USER_MAC_ADDRESS—MAC address of DHCP subscriber session
- PA_USER_SESSION_ID—RADIUS session ID for the subscriber session
- PA_USER_TYPE—Type of subscriber session: ASSIGNEDIP, AUTHINTF, INTF, ADDR, AUTHADDR, PORTAL
- PA_USER_RADIUS_CLASS—RADIUS class of the subscriber session that is associated with the service session
- STATUS—Accounting status: start, stop, and interim

- Default—STATUS,NAS_ID,PA_SSP_HOST,PA_ROUTER_NAME, PA_INTERFACE_NAME,PA_INTERFACE_ALIAS,PA_INTERFACE_DESCR, PA_PORT_ID,PA_USER_IP_ADDRESS,PA_LOGIN_NAME,PA_ACCOUNTING_ID, PA_AUTH_USER_ID,PA_IF_RADIUS_CLASS,PA_IF_SESSION_ID, PA_SERVICE_NAME,PA_RADIUS_CLASS,PA_EVENT_TIME,PA_SESSION_ID, PA_TERMINATE_CAUSE,PA_SESSION_TIME,PA_IN_OCTETS,PA_OUT_OCTETS, PA_IN_PACKETS,PA_OUT_PACKETS,PA_NAS_IP,PA_USER_MAC_ADDRESS, PA_SERVICE_SESSION_NAME,PA_SERVICE_SESSION_TAG,PA_USER_TYPE, PA_USER_RADIUS_CLASS,PA_USER_SESSION_ID
- Property name—Fields

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

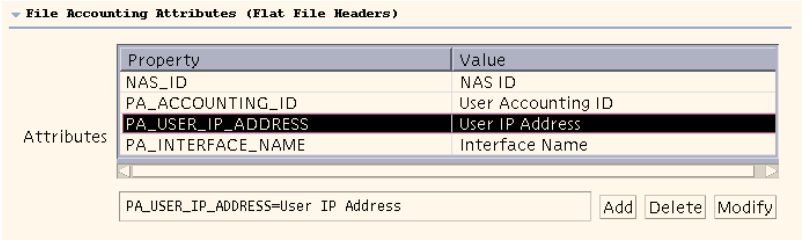
start,SSP.uelmo,uelmo,default@erx7_ssp57,FastEthernet1/1.1,,IP1/1.1,default@erx7_ssp57 FastEthernet1/1:65535, 10.10.10.20,pebbles@virneo.net,,,,erx fastEthernet 1/1:0001048619,Video-Gold,Video-Gold,Fri Jan 30 14:23:29 EDT 2004, VideoGold:null:1064946209182, 0,0,0,0,0,0, 10.10.7.17,,,,PPP,, pebbles:1064946144841

You can assign your own names to the headers that appear in the file. To do so, you 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.

To set up a file accounting template:

1. In the File-Acct Template tab, create a File Accounting Attributes instance as described in *Creating Plug-In Instances* on page 143.

   The new instance appears.



2. Define header names in the attribute table in the format property = value, where property is the attribute name and value is the header name that you want to assign to the attribute. Configure the attribute table as follows:

   - To add an attribute, type the attribute definition in the format property = value in the field below the attribute table, and click **Add**.
   - To modify an attribute, select the attribute, make your changes in the field below the attribute table, and click **Modify**.
   - To delete an attribute, select the attribute, and click **Delete**.

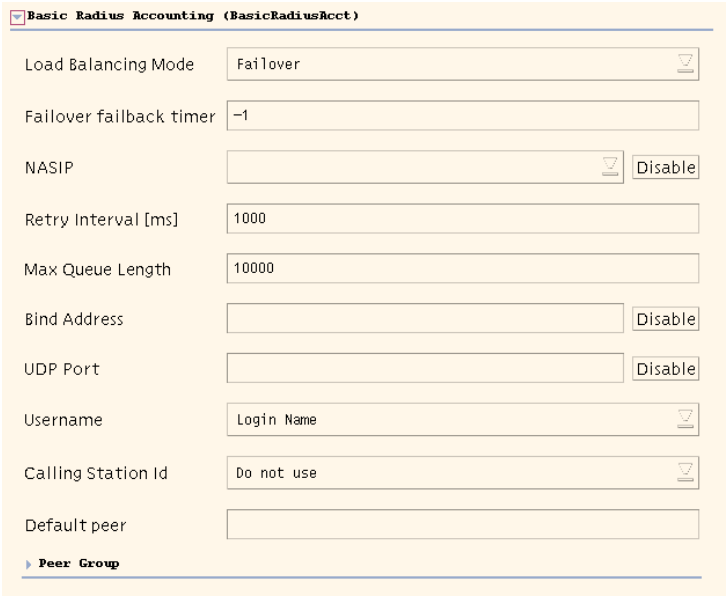
Configuring Basic RADIUS Accounting Plug-Ins

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.

To set up basic RADIUS accounting plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create a basic RADIUS accounting plug-in instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the fields for the plug-in instance as described in *Using RADIUS Plug-In Fields* on page 170.

3. In the Peer Group area, create at least one RADIUS peer to use as the default peer. See *Creating RADIUS Peers* on page 175.

Configuring Flexible RADIUS Accounting Plug-Ins

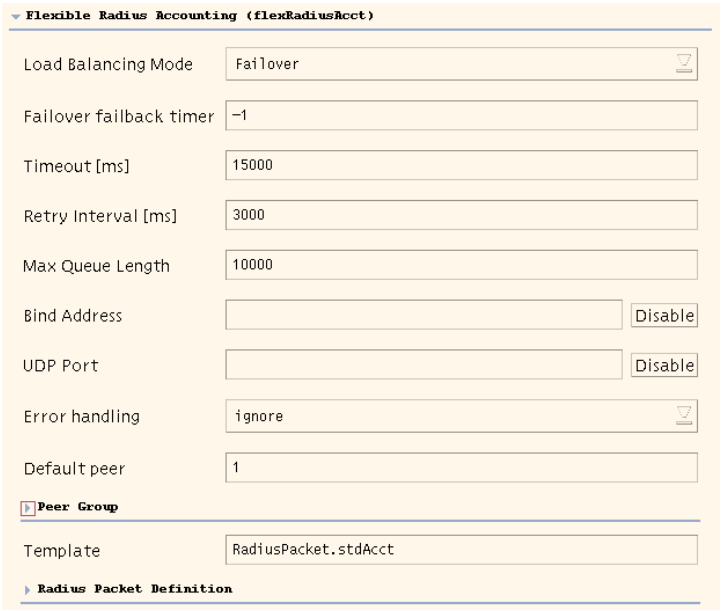
Flexible RADIUS accounting plug-ins provide the same features as basic RADIUS accounting plug-ins. In addition, they allow you to customize RADIUS accounting packets that the SAE sends to RADIUS servers. You can specify which fields are included in the RADIUS accounting packets and what information is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

You can also extend custom RADIUS plug-ins to perform the same functions as the flexible RADIUS plug-ins. These custom plug-ins are also internal plug-ins, but are designed to deliver better system performance. See *Configuring Custom RADIUS Accounting-Plug-Ins* on page 159.

To set up flexible RADIUS accounting plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create a flexible RADIUS accounting plug-in instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the fields for the plug-in instance as described in *Using RADIUS Plug-In Fields* on page 170.

3. In the Peer Group area, create at least one peer to use as the default peer. See *Creating RADIUS Peers* on page 175.

4. (Optional) Assign a RADIUS packet template to the instance, or create a packet definition for the instance. See *Defining RADIUS Packets for Flexible RADIUS Plug-Ins with SDX Configuration Editor* on page 176.

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 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 in the SAE Core API documentation on the Juniper Networks Web site at

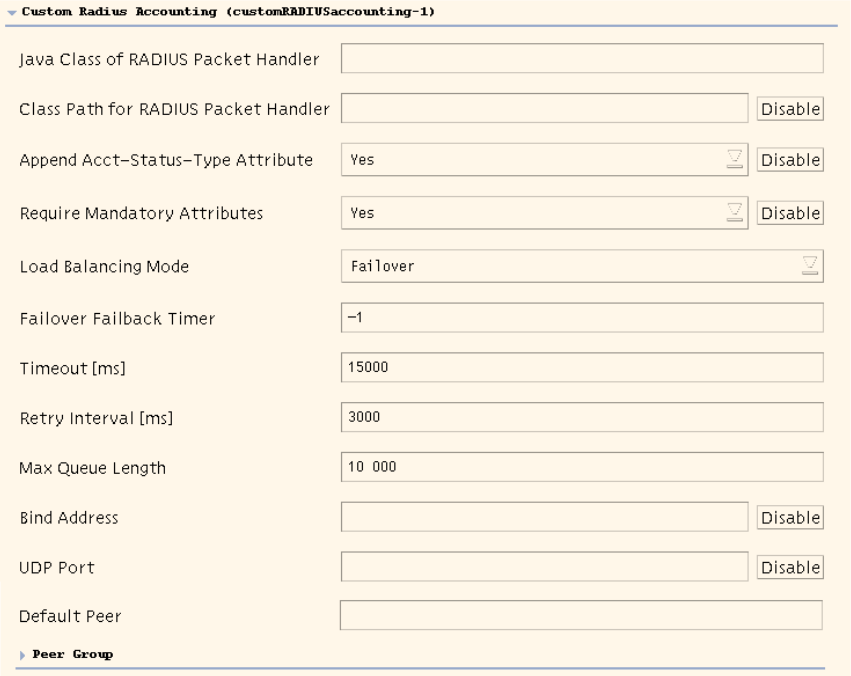
http://www.juniper.net/techpubs/software/management/sdx/api-index.html

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

To set up custom RADIUS accounting plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create a custom RADIUS accounting plug-in instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the plug-in instance fields as described in *Using RADIUS Plug-In Fields* on page 170.

3. In the Peer Group area, create at least one peer to use as the default peer. See *Creating RADIUS Peers* on page 175.

Configuring Authorization Plug-Ins

This section shows how to configure the authorization 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 authorization 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.) Authorization succeeds if all plug-in calls succeed.

Table 17: Authorization 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
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

The overall steps to configure an authorization plug-in are:

1. Create and configure a plug-in instance in the plug-in pool. The following sections show how to create and configure an instance for each type of authorization plug-in.

2. Configure an event publisher to publish events to the plug-in instance.

   See *Configuring Event Publishers* on page 184.

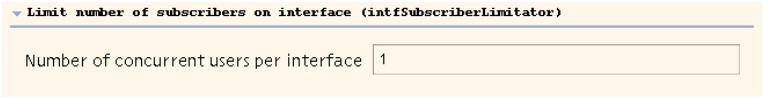
Limiting Subscribers on Router Interfaces

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.

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

1. In the Plug-In Pool area of the Plug-Ins pane, create a Limit number of subscribers on interface plug-in instance as described in *Creating Plug-In Instances* on page 143.

The instance appears in the Plug-In Pool area.



2. Fill in the number of authenticated subscribers that you want connected to an interface simultaneously.

Number of concurrent users per interface

- Number of authenticated subscribers who can connect to an IP interface on the router simultaneously.
- Value—Integer in the range 0–2147483647
- Default—1
- Property name—max_user

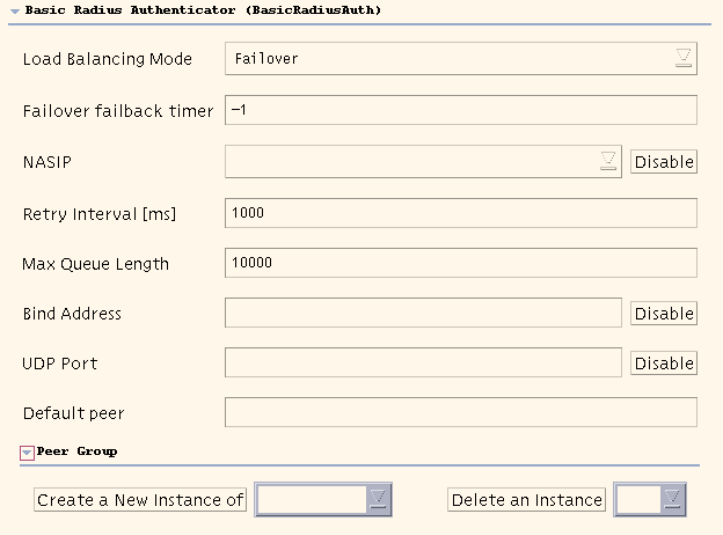
Configuring Basic RADIUS Authentication Plug-Ins

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.

To set up basic RADIUS authentication plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create a basic RADIUS authentication plug-in instance as described in *Creating Plug-In Instances* on page 143.

The instance appears in the Plug-In Pool area.



2. Fill in the fields for the plug-in instance as described in *Using RADIUS Plug-In Fields* on page 170.

3. In the Peer Group area, create at least one RADIUS peer to use as the default peer. See *Creating RADIUS Peers* on page 175.

Configuring Flexible RADIUS Authentication Plug-Ins

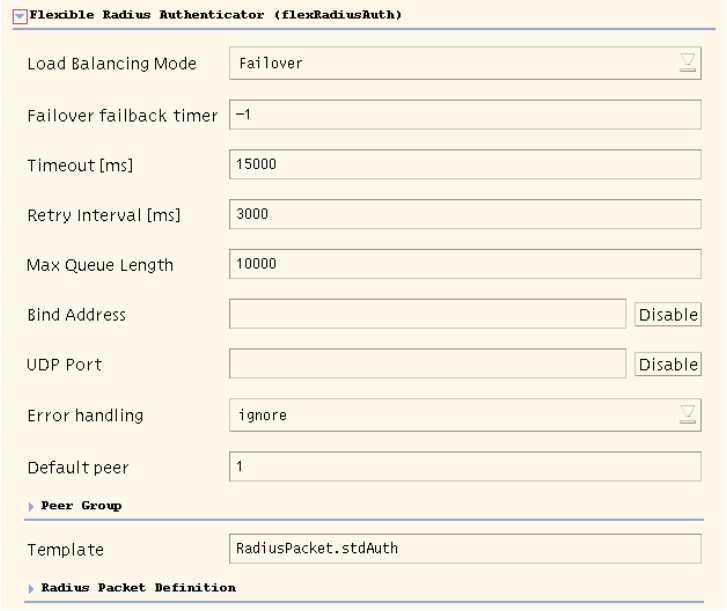
Flexible RADIUS authentication plug-ins provide the same features as basic RADIUS authentication plug-ins. In addition, they allow you to customize RADIUS authentication packets that the system sends to RADIUS servers and specify which fields are included in the RADIUS authentication packets and what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

You can also extend custom RADIUS plug-ins to perform the same functions as the flexible RADIUS plug-ins. These custom plug-ins are also internal plug-ins, but are designed to deliver better system performance. See *Configuring Custom RADIUS Authentication Plug-Ins* on page 164.

To set up flexible RADIUS authentication plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create a flexible RADIUS authentication plug-in instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the plug-in instance fields as described in *Using RADIUS Plug-In Fields* on page 170.

3. In the Peer Group area, create at least one peer to use as the default peer. See *Creating RADIUS Peers* on page 175.

4. (Optional) Assign a RADIUS packet template to the instance, or create a packet definition for the instance. See *Defining RADIUS Packets for Flexible RADIUS Plug-Ins with SDX Configuration Editor* on page 176.

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 which 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

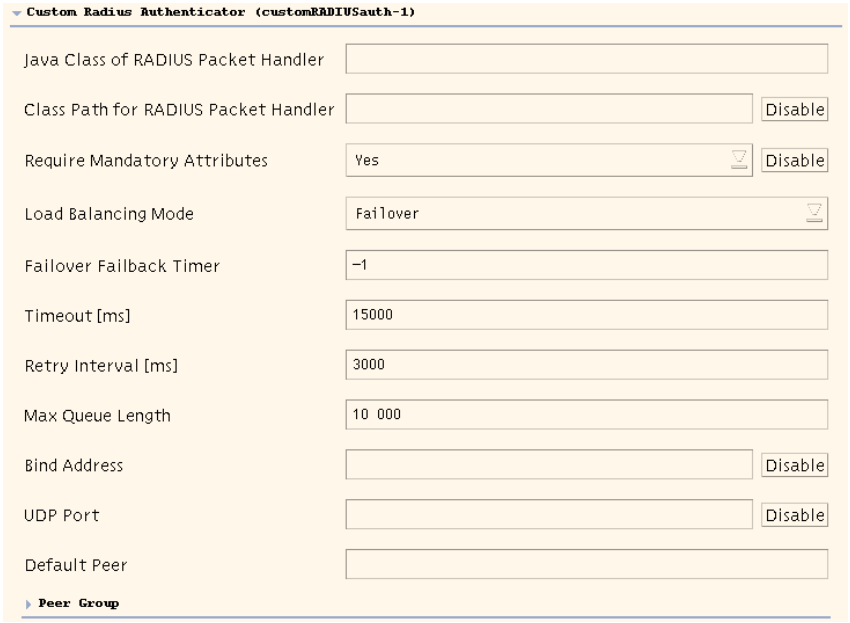
http://www.juniper.net/techpubs/software/management/sdx/api-index.html

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

To set up custom RADIUS authentication plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create a custom RADIUS authentication plug-in instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the plug-in instance fields as described in *Using RADIUS Plug-In Fields* on page 170.

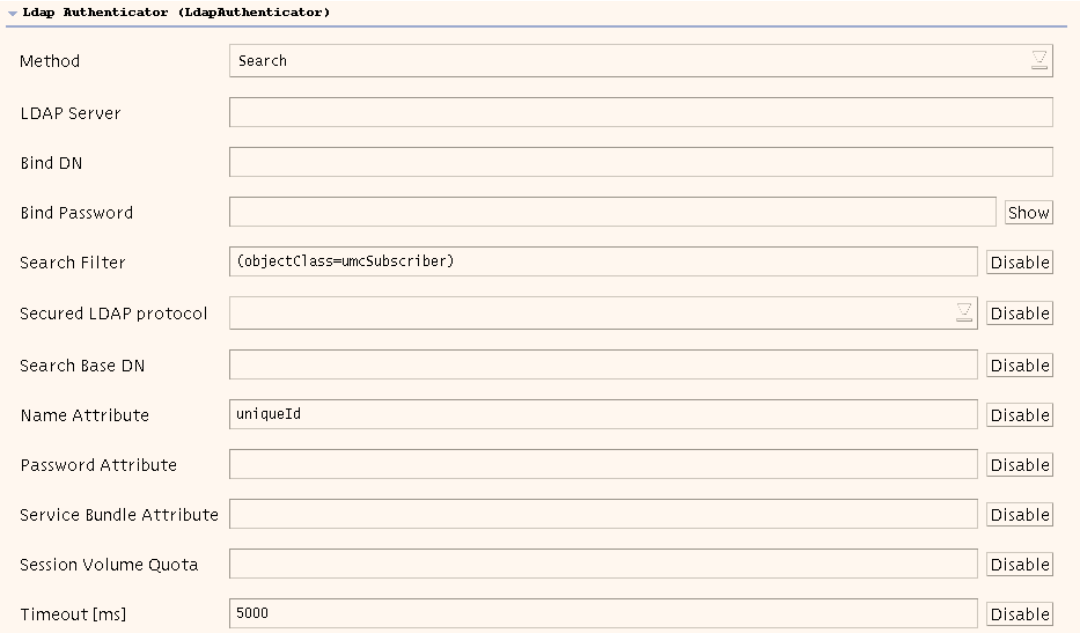
3. In the Peer Group area, create at least one peer to use as the default peer. See *Creating RADIUS Peers* on page 175.

Configuring LDAP Authentication Plug-Ins

To create LDAP authentication plug-ins:

1. In the Plug-In Pool area of the Plug-Ins pane, create an Ldap authenticator plug-in instance as described in *Creating Plug-In Instances* on page 143.

   The instance appears in the Plug-In Pool area.



2. Fill in the plug-in instance fields as described below.

Method

- LDAP authentication method that the SAE uses.
- Value
  - search—SAE searches the directory for the username that the subscriber enters, retrieves the found object, and compares the password stored in the object with the provided password.

    You can store passwords in clear text or encrypted (hashed) format by using the crypt (UNIX /etc/passwd), SHA, or MD5 algorithms. The format for a hashed password is:

    {crypt}*hashed password*, {sha}*base64 SHA password*, or {md5}*base64 MD5 password*.

  - bind—SAE performs a directory search, retrieves the DN of the found object, and tries to bind this DN and the password that the subscriber provides.

    If you specify the bind method, the plug-in uses the provided username and password to authenticate the directory (bind).

    You can store passwords in clear text or encrypted (hashed) format by using the crypt (UNIX /etc/passwd), SHA, or MD5 algorithms. You must use an encryption method that the directory supports.

- Guidelines—Both search and bind have different implications for system security and performance. When you design the system, consider:
  - search—Because the SAE retrieves passwords from the directory, the directory must allow read access to the password. Allowing read access can be a security risk because an attacker may be able to read passwords in subscriber profiles. However, to lower the risk of password exposure, you can store passwords in encrypted (hashed) form.
  - bind—SAE sends the password to the directory for authentication. The advantage is that passwords never need to be read from the directory. However, passwords are sent in clear text, and an attacker could intercept them.

    Bind is a relatively expensive operation that can affect system performance.
- Default—search
- Property name—method

LDAP Server

- Comma-separated list of IP addresses or hostnames of the LDAP authentication server.
- Value—IP address
- Default—127.0.0.1
- Property name—host

Bind DN

- DN used to authenticate access to the directory.
- Value—DN
- Default—*cn = ssp, ou = Components, o = Operators, < base >*
- Property name—bindDN

Bind Password

- 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.
- Value—Characters that make up the password; SDX Configuration Editor encodes the secret using base64
- Default—ssp
- Property name—bindPW

Search Filter

- Additional LDAP search filter that the SAE uses to search the directory for the subscriber profile. The initial search uses a search filter in the form (&(*nameAttribute = userName*) *filter*). The search is successful when the username and the filter match.
- Value—Search filter syntax defined in RFC 2254—The String Representation of LDAP Search Filters (December 1997)
- Default—(objectClass = umcSubscriber)
- Property name—filter

Secured LDAP protocol

- Secure protocol used for LDAP connections with the directory. LDAPS, the only protocol supported, causes communication with the directory to be encrypted with Secure Sockets Layer (SSL).
- Value—LDAPS
- Default—LDAPS
- Property name—securityProtocol

Search Base DN

- Base DN for searching entries in the directory. If you do not specify a base DN, the SAE uses the DN of the associated retailer object.
- If you do not specify the base DN, the SAE takes a username in the form subscriber@domain and maps domain to a retailer object by comparing *domain* with the domain names stored in the retailer object. There are two special cases:
  - If domain is empty, first the virtual router name and then the name default are tried.
  - If a retailer defines * (asterisk) as a domain name, it is used to map all domains that cannot be mapped directly.

- Value—DN
- Default—No value
- Property name—baseDN

Name Attribute

- Name of the directory attribute that holds the username.
- Value—Attribute name
- Default—uniqueID
- Property name—nameAttr

Password Attribute

- Name of the directory attribute that stores the password.
- Value—Directory attribute name
- Default—userPassword
- Property name—pwdAttr

Service Bundle Attribute

- 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.
- Value—Directory attribute name
- Default—No value
- Property name—serviceBundleAttr

Session Volume Quota

- 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.
- Value—Name of LDAP attribute.
- Default—No value
- Property name—sessionVolumeQuotaAttr

Timeout

- Maximum time the SAE waits for a response from a directory server. If the directory server does not respond to the request, the request fails and the SAE logs an error message.
- Value—Number of milliseconds in the range 0–2147483647
- Default—5000
- Property name—operationTimeout

Using RADIUS Plug-In Fields

This section describes the fields in RADIUS plug-ins.

Append Acct-Status-Type Attribute

- Specifies whether or not the plug-in includes the Acct-Status-Type attribute in a RADIUS accounting request packet.
- Values—Yes or No
- Default—Yes
- Property name —setAcctStatusType

Bind Address

- Source IP address that the plug-in uses to communicate with the RADIUS server.
- Value—IP address; if you do not specify an address, the global default address is used. The SAE automatically sets the global default address when you run the **etc/config** command during initial configuration of the SAE. The property for the global address is the AccountingMgr.local.address property in the */opt/UMC/sae/etc/default.properties* file.
- Default—No value
- Property name—local.address

Calling Station Id

- Specifies whether the SAE sends the MAC address of the subscriber in the Calling-Station-Id attribute.
- Value—Send Mac address or Do not use
- Default—Do not use
- Property name—CallingStationId

Class Path for RADIUS Packet Handler

- List of URLs that identify a location from which Java classes are loaded when the plug-in is initialized. Commas separate each URL in the list.
- Value— < class path >
- Guideline—If no value is specified, the SAE loads Java classes specified in the class path for the SAE, including the */opt/UMC/sae/lib* directory.
- Default—No value
- Property name —handler.classpath

Default peer

- Name of the RADIUS server to which the SAE sends accounting packets.
- Value—Name of the server as defined in the RADIUS peer configuration
- Default—No value
- Property name—defaultPeer

Error handling

- Configures the way the SAE handles errors.
- Value
  - ignore—Ignores incorrect definitions and logs them for debugging purposes
  - strict—Logs errors and discards the affected RADIUS packet
- Default—ignore
- Property name—attr_error

Failover failback timer

- Controls if and when the SAE attempts to fail back to the default peer.
- Value—Integer
  - Number of seconds in the range 1–2147483647 after a failover that the SAE attempts to fail back
  - 0—SAE always attempts to fail back
  - –1—SAE never attempts to fail back
- Default— –1
- Property name—failbackTimer

Java Class of RADIUS Packet Handler

- Name of the Java class that implements the RadiusPacketHandler interface in the RADIUS Client Library.
- Value— < class name >
- Default—No value
- Example—net.juniper.smgt.radius.RadiusPacketHandlerImpl
- Property name —handler.class

Load Balancing Mode

- Selects the mode for load-balancing RADIUS servers.
- Value—Failover, round-robin
  - Failover—SAE sends requests to the RADIUS server configured as the default peer. If the default peer fails, the SAE uses the next server configured in the peer group. The SAE cycles through the configured RADIUS servers as needed.
  - Round-robin—SAE alternates requests between all RADIUS servers configured in the peer group.
- Default—Failover
- Property name—loadBalancingMode

Max Queue Length

- Maximum number of unacknowledged RADIUS messages that the plug-in receives from the RADIUS server before it discards new messages.
- Value—Integer in the range 0–2147483647
- Default—10000
- Property name—local.maxWaitingQueueLength

NASIP

- Value of the NAS-Ip attribute.
- Value—SSP local IP, RADIUS client IP
  - SSP local IP—IP address of the SAE
  - RADIUS client IP—IP address of the virtual router
- Default—No value
- Property name—local.NASIP

Require Mandatory Attributes

- Specifies whether or not a RADIUS authentication or accounting request must contain all mandatory RADIUS attributes before sending the request packet.
- Values—Yes or No
- Default—Yes
- Property name—forceMandatoryAttr

Retry interval [ms]

- Time the SAE waits for a response from a RADIUS server before it resends the RADIUS packet. The SAE keeps sending RADIUS packets until either the server acknowledges the packet or the maximum timeout is reached.
- Value—Number of milliseconds in the range 0–2147483647
- Default—3000
- Property name—local.retryInterval

Template

- Name of a template that defines sets of RADIUS attributes included in accounting messages. You define templates in the RADIUS tab of SDX Configuration Editor. See *Defining RADIUS Packets for Flexible RADIUS Plug-Ins with SDX Configuration Editor* on page 176.
- Value—Name of the template in the format RadiusPacket.<template name>; you can enter only one template name
- Default—RadiusPacket.sdtAcct
- Property name—RadiusPacket.<template name>

Timeout [ms]

- Maximum time the SAE waits for a response from a RADIUS server. If the RADIUS server does not respond to the request, the request fails and the SAE logs an error message.
- Value—Number of milliseconds in the range 0–9223372036854775807
- Default—10000
- Property name—local.timeout

UDP Port

- Source UDP port or a pool of ports that the plug-in uses to communicate with the RADIUS server.
- Value—You can enter a single port number, a pool of port numbers, or a list of port numbers and port ranges. If you do not specify a UDP port, the global default port is used (see *Configuring UDP Ports for RADIUS Plug-Ins* on page 174*).*
  - Port number in the range 1–65535
  - A range of ports in the format port-port; for example, 7000-7003
  - A comma-separated list of port numbers and port ranges
- Default—No value
- Example—7000-7003, 7006, 7007-7009
- Property name—local.port

Username

- Value of the User-Name attribute (RADIUS attribute [1]).
- Value—One of the following:
  - Login Name—Name used for login
  - Accounting ID—Value stored in the subscriber profile
  - Auth User Name—Name used to authenticate a service
  - Manager ID—Value of the manager ID in the service subscription; use this setting to identify subscribers to enterprise services. Manager ID is the value of modifiersName in the subscription; if modifiersName does not exist, manager ID is the value of creatorsName.
    - modifiersName—Contains DN of the administrator who last modified the entry in the directory
    - creatorsName—Contains DN of the administrator who created the entry in the directory
- Default—Login Name
- Property name—Username

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.

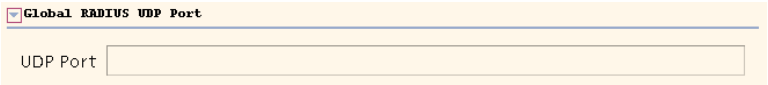
To overcome this limitation, you can configure a pool of UDP ports for RADIUS plug-ins. Having a pool of ports allows RADIUS plug-ins to create one queue per port to wait for RADIUS replies. Each queue can wait for 256 RADIUS packets. The RADIUS plug-ins send RADIUS packets through the pool of ports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

To configure global UDP ports with SDX Configuration Editor:

1. In the navigation pane, select a directory configuration object for the SAE that you want to configure.
2. Select the **Miscellaneous** tab, and expand the **Global RADIUS UDP Port** section.



3. Fill in the field as described in *Global RADIUS UDP Port Field* on page 174.

Global RADIUS UDP Port Field

Use the field in this section to specify a global UPD port for RADIUS plug-ins.

UDP Port

- Source UDP port or a pool of ports that RADIUS plug-ins use to communicate with RADIUS servers.
- Value—You can enter a single port number, a pool of port numbers, or a list of port numbers and port ranges.
  - Port number in the range 1–65535
  - A range of ports in the format port-port; for example, 7000-7003
  - A comma-separated list of port numbers and port ranges

- Default—18130
- Example—7000-7003, 7006, 7007-7009
- Property name—AccountingMgr.local.port

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.

NOTE: If you configure more than one RADIUS peer in a plug-in instance that has the same properties, the SNMP counters for the plug-in will not update correctly. The reason is that the software does not know which RADIUS peer to send updates to.

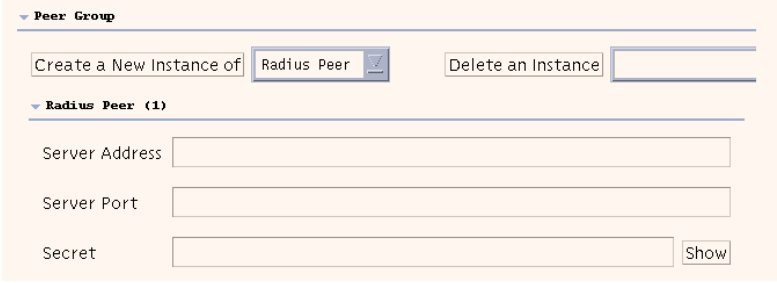
RADIUS peers are configured in the peer group for each RADIUS plug-in instance. To create a RADIUS peer:

1. In the Peer Group area of a RADIUS plug-in instance, select **Radius Peer** and click **Create a New Instance of**.

   The Create New Instance dialog box appears.

2. Assign a name to the instance, and click **OK**.

   The new peer instance appears in the Peer Group area.



3. Fill in the fields as described below.

Server Address

- IP address of the RADIUS server to which the SAE sends accounting data.
- Value—IP address
- Default—No value
- Property name—peer.#.remote.address

Server Port

- Port used for RADIUS accounting packets. RADIUS accounting servers typically use UDP port 1813 or 1646.
- Value—Valid UDP port
- Default—1813
- Property name—peer.#.remote.port

Secret

- Password that is shared with the RADUS server. You must configure the same secret on the RADIUS server.
- Value—Shared secret; SDX Configuration Editor encodes the secret using BASE-64
- Default—No value
- Property name—peer.#.remote.password

Defining RADIUS Packets for Flexible RADIUS Plug-Ins with SDX Configuration Editor

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.

In SDX Configuration Editor, 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-in instances. You can apply the same template to multiple plug-in instances, but each plug-in instance can use only one template.
- Define attributes in the packet definition configuration of a flexible plug-in instance. These definitions override definitions in packet templates. You can use these packet definitions to exclude attributes that come from the template. To do so, you define the value of the attribute that you want to exclude as None.

Creating and Using RADIUS Templates

The SDX 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

You can use these templates as they are, modify them, or create new templates.

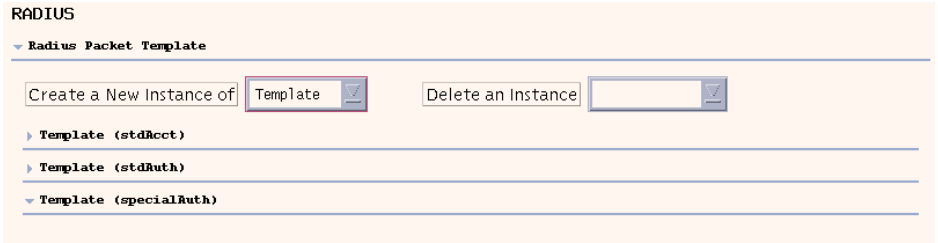
To create a template:

1. In the RADIUS tab, select **Template** from the drop-down list, and click **Create a New Instance of**.

   The Create a New Instance dialog box appears.

2. Assign a name to the template instance, and click **OK**.

   The instance appears in the Radius Packet Template area.



3. Configure RADIUS attributes in the template as described in the next section.

4. Configure a plug-in instance to use the template by entering the name of the template in the format RadiusPacket.<template name> in the Template field of the plug-in instance configuration.

   You can apply a template to multiple plug-in instances, but each plug-in instance can use only one template.

Configuring RADIUS Attributes

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
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

Table 18: RADIUS Attribute Instance Names (continued)

Attribute Instance (Packet-Type)	Type of RADIUS Packet to Which Attribute Definition Is Applied
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	User Session Start, Stop, or Interim requests
userresp	Any user authorization response
userstart	User Session Start requests
userstop	User Session Stop, or Interim requests

Use the steps below to configure attribute instances. You can follow them from within a RADIUS template or within a plug-in instance configuration.

You can configure attribute instances in a RADIUS template or within a plug-in instance configuration. To create and configure attribute instances for a:

- Template—Follow these steps in the Attributes configuration section of a template.
- Plug-in instance—Follow these steps in the Radius Packet Definition of a plug-in instance.

1. Select **Radius Attributes** from the drop-down list, and click **Create a New Instance of**.

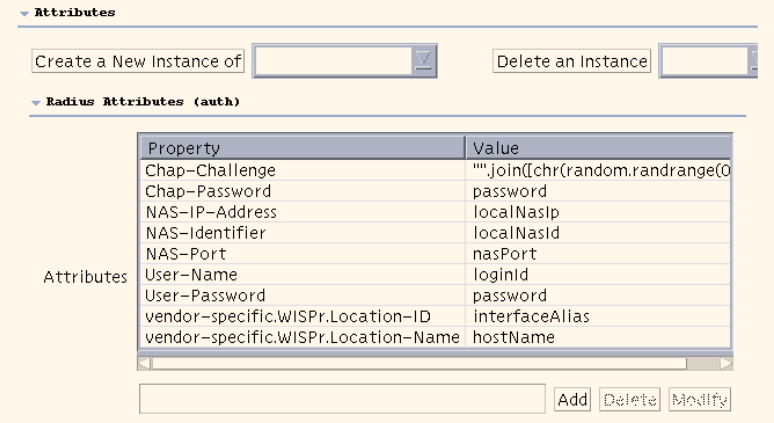
   The Create a New Instance dialog box appears.

2. Assign a name that specifies the RADIUS packet type to which the attribute definition applies (see Table 18), and click **OK**.

   A new attribute table of properties (RADIUS attributes) and values (the value assigned to an attribute) appears.

3. Configure the attribute table as follows:

   - To add an attribute, type the attribute definition in the format property = value in the field below the attribute table, and click **Add**.
   - To modify an attribute, select the attribute, make your changes in the field below the attribute table, and click **Modify**.
   - To delete an attribute, select the attribute, and click **Delete**.



Property

- RADIUS attribute.
- Value—Standard RADIUS attribute or JUNOSe VSA specified as follows:
  - Standard RADIUS attribute 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
  - JUNOSe VSA in 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

  For example, 26.4874.50.text sets a value for Session-Volume-Quota VSA 26-50.
- Default—No value
- Property name— < id > [. < type >]

Value

- Defines the values of RADIUS attributes. Most values can be sent from the SAE to the plug-in. Some of the values can also be returned by the plug-in.
- Value—Standard values (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.
- Default—No value
- Property name— < expression >

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 14 on page 113 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.

Table 19: Standard Values for RADIUS Attributes (continued)

Value	Type of Plug-In	Comments
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.
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	

Table 19: Standard Values for RADIUS Attributes (continued)

Value	Type of Plug-In	Comments
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

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:

  ```
  RadiusPacket.std.svcstop.Acct-Input-Octets = lowWord(inOctets)
  RadiusPacket.std.svcstop.Acct-Output-Octets = lowWord(outOctets)
  RadiusPacket.std.svcstop.Acct-Input-Packets = inPackets
  RadiusPacket.std.svcstop.Acct-Output-Packets = outPackets

  RadiusPacket.std.svcstop.Acct-Input-Gigawords = highWord(inOctets)
  RadiusPacket.std.svcstop.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 180 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.

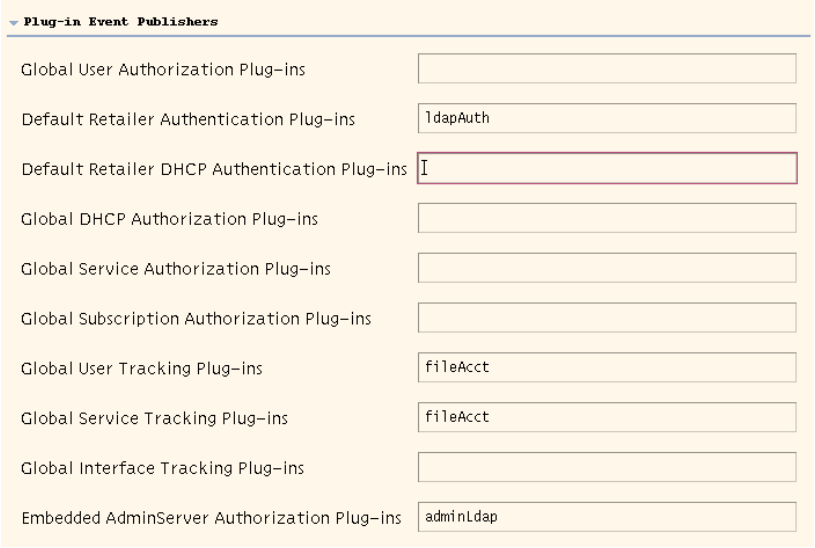
Configuring Event Publishers

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

Configuring Global and Default Retailer Event Publishers

You can use SDX Configuration Editor to configure global and default retailer event publishers. To do so:

1. Access the plug-in configuration as described in *Accessing the Plug-In Configuration* on page 141.



2. In the Plug-In Event Publishers area, enter a comma-separated list of plug-in instances in each event publisher field that you want to configure.

Global User Authorization Plug-ins

- Authorize all subscriber sessions. These plug-in instances are called after a subscriber profile is loaded but before a subscriber session is started. The SAE calls these plug-ins for each subscriber who logs in to a portal.
- These plug-in instances cannot perform authentication, because passwords are not available at this point in the login process. Therefore, if you specify plug-in instances that perform authentication, login requests will fail.
- Value—Comma-separated list of plug-in instances
- Default—No value
- Property name—User.auth.plugins

Default Retailer Authentication Plug-ins

- Authenticate subscribers who are assigned to retailer objects that do not specify an authentication plug-in. These plug-ins are called when a subscriber logs in to the domain. The authentication process for portal (Web) logins maps the supplied domain name to a retailer object.
- If you do not specify default retailer authentication plug-ins or retailer-specific plug-ins, subscribers are admitted without authentication.
- Value—Comma-separated list of plug-in instances

- Default—ldapAuth
- Property name—Retailer.auth.plugins

Default Retailer DHCP Authentication Plug-ins

- Authenticate DHCP address requests for subscribers who are assigned to retailer objects that do not specify a DHCP authorization plug-in. These plug-ins are called when the SAE receives a DHCP discover request from a client that has its username and password cached in the SAE. The username and password can either be cached persistently in the directory or temporarily in memory during a switch from an unauthenticated to an authenticated address.
- Value—Comma-separated list of plug-in instances
- Default—No value
- Property name—Retailer.dhcpauth.plugins

Global DHCP Authorization Plug-ins

- Authorize all DHCP address requests for all DHCP subscribers who log in to a portal. These plug-in instances are called for both authenticated and unauthenticated address requests.
- Value—Comma-separated list of plug-in instances
- Default—No value
- Property name—Dhcp.auth.plugins

Global Service Authorization Plug-ins

- Authorize all service sessions. These plug-ins are called before a service session is started, and are called for every service session started by any SAE subscriber.
- Value—Comma-separated list of plug-in instances
- Default—No value
- Property name—Service.auth.plugins

Global Subscription Authorization Plug-ins

- Authorize subscribers to change their subscriptions. These plug-ins are called when a subscriber tries to modify, subscribe to, or unsubscribe from a subscription.
- Value—Comma-separated list of plug-in instances
- Default—No value
- Property name—Subscription.auth.plugins

Global User Tracking Plug-ins

- Track all subscriber sessions. These plug-in instances are called for every subscriber session that is started and stopped. They are called after a subscriber session has started and when the session is stopped.
- Value—Comma-separated list of plug-in instances

- Default—fileAcct
- Property name—User.tracking.plugins

Global Service Tracking Plug-ins

- Track all service sessions. These plug-in instances are called for every service session that is started and stopped. They are called after a service session starts, when the service session stops, and during interim updates.
- Value—Comma-separated list of plug-in instances
- Default—fileAcct
- Property name—Service.tracking.plugins

Global Interface Tracking Plug-ins

- Track all interfaces that the SAE manages. You can set up the publisher to send interface tracking events to plug-in instances or to a network information collector (NIC) SAE plug-in agent. These plug-in instances and/or NIC SAE plug-in agents are called for every managed interface that 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.
- Value—Comma-separated list of plug-in instances or NIC SAE plug-in agents
- Default—No value
- Property name—Interface.tracking.plugins

Embedded AdminServer Authorization Plug-ins

- Authorize administrators to connect to the embedded Web server, which is used to access SAE Web Admin.
- Value—Comma-separated list of plug-in instances
- Default—adminLdap
- Property name—AdminServer.realm.auth.plugins

Configuring Service-Specific Event Publishers

In the value-added services definition in SDX Admin, 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.

You configure these event publishers in the SSP Services window in SDX Admin. See *SRC-PE Services and Policies Guide, Chapter 2, Managing Services on a Solaris Platform*.

Configuring Retailer-Specific Event Publishers

In the retailer definition in SDX Admin, 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.

You configure these event publishers in the Retailer pane in SDX Admin. See *Adding Retailers* on page 236.

Configuring Virtual Router–Specific Event Publishers

In the virtual router definition in SDX Admin, 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.

You configure this event publisher in the VirtualRouter pane in SDX Admin. For information about configuring virtual routers for JUNOSe routers, see *SRC-PE Network Guide, Chapter 6, Using JUNOSe Routers in the SRC Network with a Solaris Platform*. For information about configuring virtual routers for JUNOS routing platforms, see *SRC-PE Network Guide, Chapter 8, Using JUNOS Routing Platforms in the SRC Network with a Solaris Platform*.