Chapter 2

Integrating Third-Party Network Devices into the SRC Network on a Solaris Platform

This chapter describes how to integrate third-party network devices into the SRC network with the SRC configuration applications that run only on Solaris platforms.

You can also use the CLI that runs on Solaris platforms and the C-series platform to configure the SRC system to work with third-party devices. See *Chapter 1, Integrating Third-Party Network Devices into the SRC Network with the SRC CLI*.

The chapter contains the following topics:

Overview of Integrating Network Devices into the SRC Network

You can integrate third-party routers and other network devices into your SRC network. The SAE provides a driver that you can use to integrate the SAE with a third-party device. This device driver uses the session store to store and replicate subscriber and service session data within a community of SAEs.

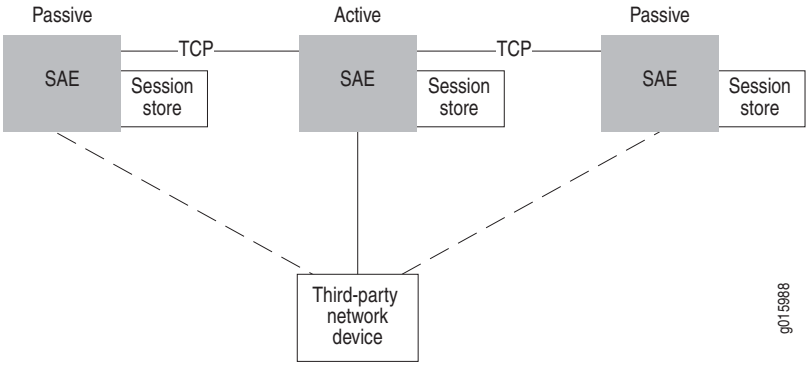
To log in subscribers to the SAE, you use assigned IP subscribers or event notification from an IP subscribers.

To activate services and provision policies on the device, you use script services. You can also activate aggregate services for subscribers. However, you cannot activate normal services that require policies to be provisioned on the device.

SAE Communities

For SAE redundancy in an SRC network, you can have a community of two or more SAEs. SAEs in a community are given the role of either active SAE or passive SAE. The active SAE manages the connection to the network device and keeps session data up to date within the community. Figure 4 shows a typical SAE community.

Figure 4: SAE Community



When an SAE starts, it negotiates with other SAEs to determine which SAE controls the network device. The SAE community manager and members of the community select the active SAE.

A passive SAE needs to take over as active SAE in any of the following cases:

- The active SAE shuts down. In this case, the active SAE notifies the passive SAEs, and one of the passive SAEs takes over as active SAE.
- A passive SAE does not receive a keepalive message from the active SAE within the keepalive interval. In this case, the passive SAE attempts to become the active SAE.

Storing Session Data

To aid in recovering from an SAE failover, the SAE stores subscriber and service session data. When the SAE manages a network device, session data is stored locally in the SAE host's file system. The SRC component that controls the storage of session data on the SAE is called the session store. The session store queues data and then writes the data to session store files on the SAE host's disk. Once the data is written to disk, it can survive a server reboot.

For more information, see *Storing Subscriber and Service Session Data* in *SRC-PE Network Guide, Chapter 3, Configuring the SAE with SDX Configuration Editor*.

Using Script Services to Provision Third-Party Devices

You use script services to activate services and provision policies on third-party network devices. A script service is a service into which you can insert or reference a script. You write a script that will activate services and provision policies on the third-party device, and then you insert the script into the script service or reference the script in the service. When the SAE activates a service, it sends the script to the network device that the SAE is managing. You can also include an interface in the script that will cause the SAE to send authentication and tracking events when it activates, modifies, or deactivates a script service session.

The SAE core API includes two interfaces for creating a script:

- ScriptService—Defines a service provider interface (SPI) that the script service must implement. The implementation of the ScriptService interface activates the service.

- ServiceSessionInfo—Provides a callback interface into SAE and provides information about the service session to the script service.

For information about the ScriptService interface and the ServiceSessionInfo interface, see the script service documentation in the SRC software distribution in the folder *SDK/doc/sae* 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

You can write the script in Java or Jython.

Logging In Subscribers and Creating Sessions

You can use two mechanisms to obtain subscriber address requests and other information and to set up a pseudointerface on the network device. (You must choose one mechanism; you cannot mix them.):

1. Assigned IP subscriber. The SAE learns about a subscriber through subscriber-initiated activities, such as activating a service through the portal or through the SRC SOAP Gateway (SRC-SG).

   With this method, you use the assigned IP subscriber login type along with the network interface collector (NIC) to map IP addresses to the SAE.

2. Event notification from an IP address manager. The SAE learns about subscribers through notifications from an external IP address manager, such as a DHCP server or a RADIUS server.

   With this method, you use the event notification application programming interface (API). The API provides an interface to the IP address manager, and lets the IP address manager notify the SAE of events such as IP address assignments.

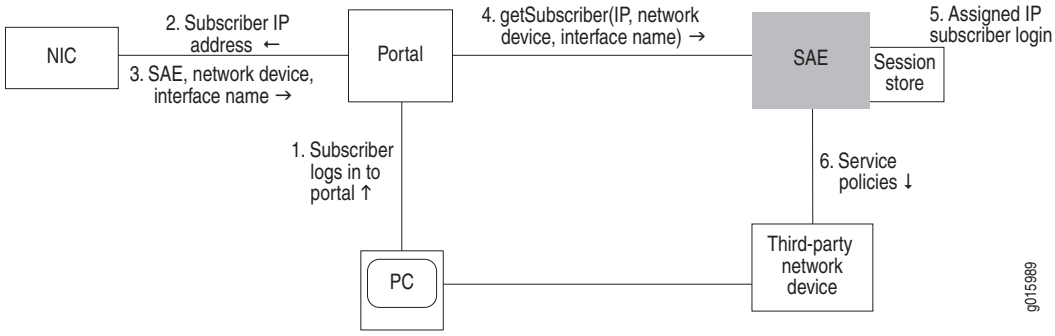
Assigned IP Subscribers

With the assigned IP subscriber method of logging in subscribers and creating sessions, the SRC software uses IP address pools along with NIC resolvers to provide mapping of IP addresses to SAEs. You configure the static address pools or dynamically discovered address pools in the virtual router configuration for a network device. These pools are published in the NIC. The NIC maps subscriber IP addresses in requests received through the portal or Advanced Services Gateway to the SAE that currently manages that network device.

Login Interactions with Assigned IP Subscribers

This section describes login interactions for assigned IP subscribers. In the example shown in Figure 5, the subscriber activates a service through a portal. You could also have the subscriber activate a service through the Advanced Services Gateway.

Figure 5: Login Interactions with Assigned IP Subscribers



The sequence of events for logging in and creating sessions for assigned IP subscribers is:

1. The subscriber logs in to the portal.
2. The portal sends the subscriber's IP address to the NIC.
3. Based on the IP address, the NIC looks up the subscriber's SAE, network device, and interface name, and returns this information to the portal.
4. The portal sends a getSubscriber message to the SAE. The message includes the subscriber's IP address, network device, and interface name.
5. The SAE creates an assigned IP subscriber and performs a subscriber login. Specifically, it
   a. Runs the subscriber classification script with the IP address of the subscriber. (Use the ASSIGNEDIP login type in subscriber classification scripts.)
   b. Loads the subscriber profile.
   c. Runs the subscriber authorization plug-ins.

d. Runs the subscriber tracking plug-ins.

e. Creates a subscriber session and stores the session data in the session store file.

6. The SAE pushes service policies for the subscriber session to the network device.

Because the SAE is not notified when the subscriber logs out, the assigned IP idle timer begins when no service is active. The SAE removes the interface subscriber session when the timeout period ends.

Event Notification from an IP Address Manager

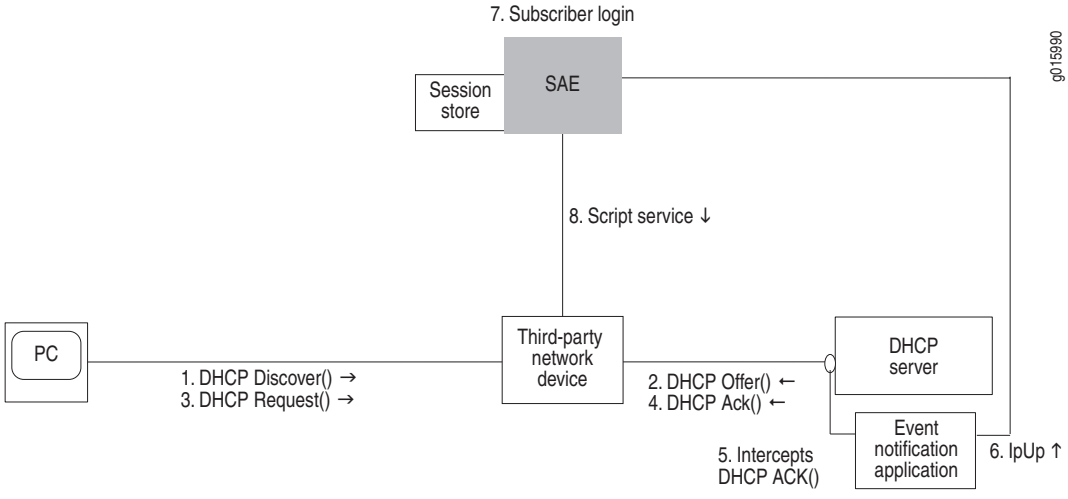
With the event notification method of logging in subscribers and creating subscriber sessions, the subscriber logs in to the network device and obtains an IP address through an address server, usually a DHCP server. The SAE receives notifications about the subscriber, such as the subscriber's IP address, from an event notification application that is installed on the DHCP server.

To use this method of logging in subscribers, you can use the event notification API to create the application that notifies the SAE when events occur between the DHCP server and the network device. You can also use Monitoring Agent, an application that was created with the event notification API, and that monitors DHCP or RADIUS messages for DHCP or RADIUS servers. See *SRC Application Library Guide, Chapter 27, Integrating IP Address Managers with the SAE*.

Login with Event Notification

This section describes login interactions using event notifications.

Figure 6: Login Interactions with Event Notification Application



The sequence of events for logging in subscribers and creating sessions is:

1. The DHCP client in the subscriber's computer sends a DHCP discover request to the DHCP server.
2. The DHCP server sends a DHCP offer to the subscriber's DHCP client.
3. The DHCP client sends a DHCP request to the DHCP server.
4. The DHCP server acknowledges the request by sending a DHCP Ack message to the DHCP client.
5. The event notification application that is running on the DHCP server intercepts the DHCP Ack message.
6. The event notification application sends an ipUp message to the SAE that notifies the SAE that an IP address is up.
7. The SAE performs a subscriber login. Specifically, it:
    a. Runs the subscriber classification script.
    b. Loads the subscriber profile.
    c. Runs the subscriber authorization plug-ins.
    d. Runs the subscriber tracking plug-ins.
    e. Creates a subscriber session and stores the session in the session store file.
8. The SAE can start script services.

The ipUp event should be sent with a timeout set to the DHCP lease time. The DHCP server sends an ipUp event for each Ack sent to the client. The SAE restarts the timeout each time it receives an ipUp event.

If the client explicitly releases the DHCP address (that is, it sends a DHCP release event), the DHCP server sends an ipDown event. If the client does not renew the address, the lease expires on the DHCP server and the timeout expires on the SAE.

Configuration Tasks for Integrating Third-Party Network Devices

To integrate third-party devices into your SRC network, you need to complete the following tasks:

- Write a script and add a script service that references the script.

  See *Setting Up Script Services* on page 26.

- Add objects for the devices to the directory.

  Adding Objects for Network Devices to the Directory on page 26.

- Set up an SAE community.

  See *Setting Up SAE Communities* on page 32.

- Configure SAE properties for the Event Notification API.

  See *Configuring SAE Properties for the Event Notification API* on page 36.

- Configure the session store.

  See *Storing Subscriber and Service Session Data* on page 41 in *SRC-PE Network Guide, Chapter 3, Configuring the SAE with SDX Configuration Editor*.

- If you are using the event notification method to log in subscribers, integrate the SAE with an IP address manager. There are two ways to do so:

  - Use the event notification API to create an application that notifies the SAE when events occur between the DHCP server and the network device.

    See the event notification API documentation in the SRC software distribution in the folder *SDK/doc/sae* or in the SAE CORBA remote API documentation on the Juniper Networks Web site at

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

  - Use Monitoring Agent, an application that was created with the event notification API, and that monitors DHCP or RADIUS messages for DHCP or RADIUS servers.

    See *SRC Application Library Guide, Chapter 27, Integrating IP Address Managers with the SAE*.

Setting Up Script Services

To set up script services:

1. Write a script that implements the ScriptService interface, a service provider interface (SPI) for the SAE.

   See *SRC-PE Services and Policies Guide, Chapter 2, Managing Services on a Solaris Platform*.

   See the script service documentation in the SRC software distribution in the folder *SDK/doc/sae* 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

2. Add a script service that references the script.

   See *SRC-PE Services and Policies Guide, Chapter 2, Managing Services on a Solaris Platform*.

Adding Objects for Network Devices to the Directory

For each network device that the SAE manages, you need to add a router object and virtual router object to the directory.

To add a router object to the directory with SDX Admin:

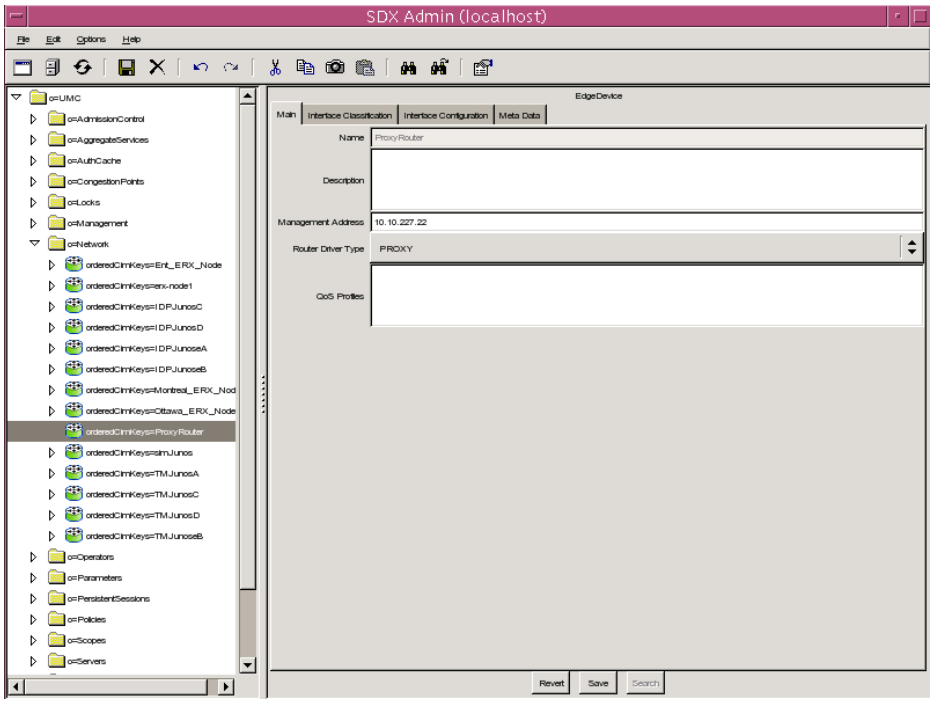
1. In the navigation pane, highlight *o = Network*, and right-click.

2. Select **New > EdgeDevice**.

   The New EdgeDevice dialog box appears.

3. In the New EdgeDevice dialog box, enter a name for the network device, and click **OK**.

The name of the new device appears in the navigation pane.



4. Use the field descriptions in *Router Fields* on page 27 to configure the router, and then click **Save**.

Router Fields

Use the fields in this section to configure routers.

Description

- Information about this device; keywords that the SDX find utility uses.
- Value—Text string
- Example—ERX-1400 router located in Ottawa

Management Address

- IP address of the network device.
- Value—IP address
- Example—192.0.1.1

Router Driver Type

- Type of device that this directory object will be used to manage.
- Value—Select PROXY
- Default—No value

QoS Profiles

- This field applies to JUNOSe routers only

Adding Virtual Router Objects

To add a virtual router object to the directory with SDX Admin:

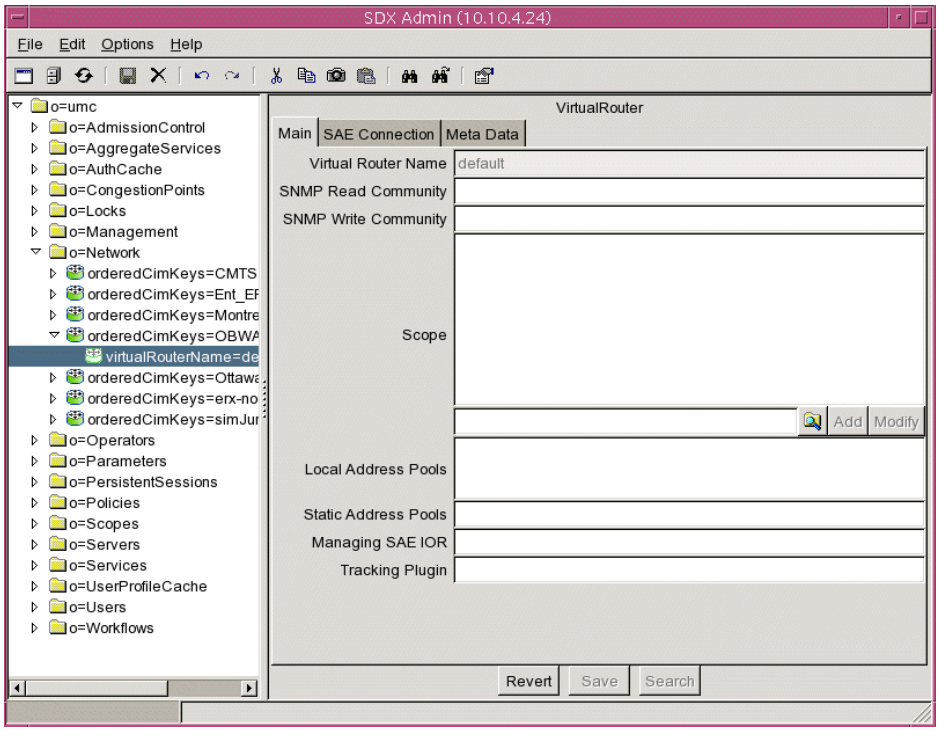
1. In the navigation pane, highlight the device to which you want to add the VR, and right-click.

2. Select **New > VirtualRouter**.

   The New VirtualRouter dialog box appears.

3. Enter the name default, and click **OK**.

   The name of the new virtual router object appears in the navigation pane, and the VirtualRouter pane appears.



4. Set the parameters in the Main tab in the VirtualRouter pane. See *Virtual Router Fields* on page 29.

5. Select the SAE Connection tab in the VirtualRouter pane, and add SAEs that are connected to the router. See *Defining SAE Communities* on page 32.

NOTE: This step is required for the SAE to work with the router.

6. Click Save in the VirtualRouter pane.

Virtual Router Fields

Use the fields in this section to define virtual routers.

SNMP Read Community

- SNMP community name associated with SNMP read-only operations for this VR.
- Value—Text string
- Example—admin

SNMP Write Community

- SNMP community name associated with SNMP write operations for this VR.
- Value—Text string
- Example—public

Scope

- Service scopes assigned to this VR—See *SRC-PE Services and Policies Guide, Chapter 2, Managing Services on a Solaris Platform*.
- Value—Text string
- Example—POP-Westford

Local Address Pools

- For JUNOSe routers only. List of IP address pools that a JUNOSe VR currently manages and stores.
- Value—You can specify an unlimited number of ranges of local IP address pools for JUNOSe VRs. You can specify either the first and last addresses in a range or the first IP address and a factor that indicates the start of the range. You can also specify IP addresses to exclude. Use spaces in the syntax only to separate the first and last explicit IP addresses in a range.

  The IP pool syntax has the format:

  ([<ipAddressStart> <ipAddressEnd>] |
  {<ipBaseAddress>/(<mask> | <digitNumber>)(,<ipAddressExclude>)*})

  - < ipAddressStart > —First IP address (version 4 or 6) in a range
  - < ipAddressEnd > —Last IP address (version 4 or 6) in a range
  - < ipBaseAddress > —Network base address

- <mask> —IP address mask
- <digitNumber> —Integer specifying the number of significant digits of the first IP address in the range
- <ipAddressExclude> —List of IP addresses to be excluded from the range
- |—Choice of expression; choose either the expression to the left or the expression to the right of this symbol
- *—Zero or more instances of the preceding group

- Guidelines—Configure this field on JUNOSe VRs only. If you do not configure the **PoolPublisher** router initialization scripts for a JUNOSe router, configure this field for the JUNOSe VR.
- Default—No value
- Example—This example shows four ranges for the IP address pool.

  ([10.10.10.5 10.10.10.250]
  {10.20.20.0/24}
  {10.21.0.0/255.255.0.0}
  {10.20.30.0/24,10.20.30.1})

  - The first range (a simple range) specifies all the IP addresses between the two IP addresses 10.10.10.5 and 10.10.10.250.
  - The second range specifies all the IP addresses in the range 10.20.20.0 to 10.20.20.255.
  - The third range uses a network mask to specify all the IP addresses in the range 10.21.0.0 to 10.21.255.255.
  - The fourth range specifies all the addresses of the network 10.20.30.0 to 10.20.30.255, excluding the address 10.20.30.1.

Static Address Pools

- For JUNOSe routers and CMTS devices only. List of IP address pools that a JUNOSe VR manages but does not store. You can configure these address pools only in the SRC software.
- Value—See the field Local Address Pools.
- Guidelines—Configure this field on JUNOSe and CMTS VRs only.
- Default—No value
- Example—([10.10.10.5 10.10.10.250] {10.20.20.0/24})

Managing SAE IOR

- Common Object Request Broker Architecture (CORBA) reference for the SAE managing this VR.
- Value—One of the following items:
  - The actual CORBA reference for the SAE
  - The absolute path to the interoperable object reference (IOR) file
  - A corbaloc URL in the form corbaloc:: < host > :8801/SAE
    - < host > is the name or IP address of the SAE host.
- Default—No value
- Guidelines—The **PoolPublisher** and **IorPublisher** router initialization scripts provide this information when the router connects to the SAE. If you do not select one of these router initialization scripts, enter a value in this field.
- Example—One of the following items:
  - Absolute path—*/opt/UMC/sae/var/run/sae.ior*
  - corbaloc URL—corbaloc::boston:8801/SAE
  - Actual IOR—IOR:000000000000002438444C3A736D67742E6A756E697...

Tracking Plug-in

- Plug-ins that track interfaces that the SAE manages on this VR. The SAE calls these plug-in instances for every interface it manages. The SAE calls these plug-ins 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
- Guidelines—Enter plug-in instances and network information collector (NIC) SAE plug-in agents that are specific to this VR.
- Default—No value
- Example—nicsae, flexRadius

Setting Up SAE Communities

You can configure the following for SAE communities:

- Define the members of an SAE community by adding the IP addresses of SAEs in the community to the virtual router object of the network device in the directory.

  See *Defining SAE Communities* on page 32.

- Configure parameters for the SAE community manager in SDX Configuration Editor.

  See *Configuring the SAE Community Manager* on page 34.

- Specify the name of the community manager.

  See *Specifying the SAE Community Manager* on page 35.

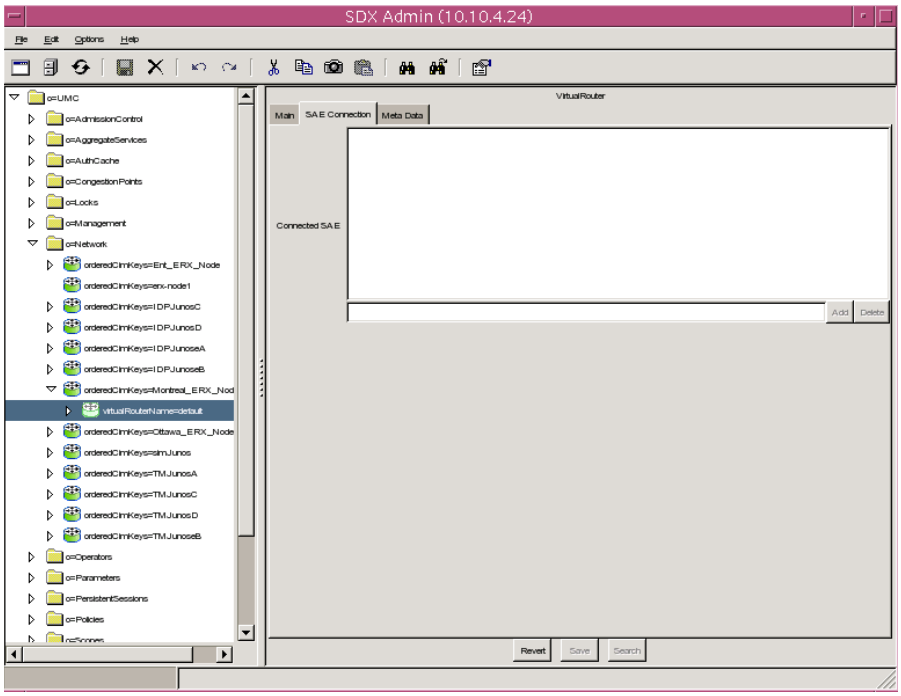
- If there is a firewall in the network, configure the firewall to allow SAE messages through.

Defining SAE Communities

You define SAE communities by entering the SAEs in a community in the connected SAE field of the virtual router object.

When you modify a community, wait for passive session stores on the new community members to be updated before you shut down the current active SAE. Otherwise, if you add a new member to a community, and then a failover from the current active SAE to the new member is triggered immediately, the new member's session store may not have received all data from the active SAE's session store.

To define a community, select the SAE Connection tab of the VirtualRouter pane in SDX Admin, and add the addresses of SAEs that can manage the device.



Adding an SAE

To add an SAE:

1. Type the IP address of the SAE in the field below the Connected SAE box.
2. Click **Add**.

Modifying an SAE Address

To modify an SAE address:

1. Click the IP address of the SAE in the Connected SAE box.
2. Modify the IP address in the field below the Connected SAE box.
3. Click **Modify**.

Deleting an SAE Address

To delete an SAE address:

1. Click the IP address of the SAE in the Connected SAE box.
2. Remove the IP address from the field below the Connected SAE box.
3. Click **Delete**.

Connected SAE

- SAEs that are connected to the network device.
- Value—IP addresses
- Default—No value

Configuring the SAE Community Manager

To use SDX Configuration Editor to configure the SAE community manager that manages third-party network device communities:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the Ext. Interface tab, and expand the Community Manager (PROXYCommunityManager) section.

3. Edit or accept the default values in the field.

   See *Community Manager Fields* on page 34.

4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Community Manager Fields

In SDX Configuration Editor, you can modify the following field in the Community Manager (PROXYCommunityManager) section of the Ext. Interface pane in an SAE configuration file.

Keepalive Interval [s]

- Interval between keepalive messages sent from the active SAE to the passive members of the community.
- Value—Number of seconds in the range 0–2147483647
- Default—30
- Property name—SAEFeature.PROXYCommunityManager.heartbeat

Number of Threads

- Number of threads that are allocated to manage the community.
- Value—Integer in the range 0–2147483647
- Guidelines—You generally do not need to change this property.
- Default—5
- Property name—SAEFeature.PROXYCommunityManager.num_threads

Acquire Timeout

- Amount of time an SAE waits for a remote member of the community when it is acquiring a distributed lock. To avoid race conditions when the SAE community is determining which SAE is the active SAE, the community manager has a distributed lock. When an SAE attempts to become the active SAE, it needs to acquire the distributed lock.
- Value—Number of seconds in the range 0–2147483647
- Guidelines—You generally do not need to change this property.
- Default—15
- Property name—SAEFeature.PROXYCommunityManager.acquire_timeout

Blackout Time

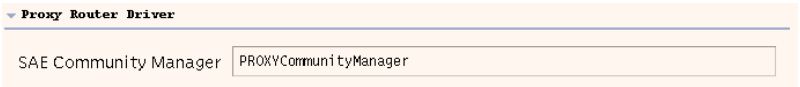
- Amount of time that an active SAE must wait after it shuts down before it can try to become the active SAE of the community again.
- Value—Number of seconds in the range 0–2147483647
- Default—30
- Property name—SAEFeature.PROXYCommunityManager.blackout_time

Specifying the SAE Community Manager

To use SDX Configuration Editor to specify the name of the community manager that manages third-party network device communities:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.

2. Select the Router tab, and expand the Proxy Router Driver section.



3. Edit or accept the default values in the field.

   See *Proxy Router Driver Fields* on page 36.

4. Select **File > Save**.

5. Right-click the configuration file, and **select SDX System Configuration > Export to LDAP Directory**.

Proxy Router Driver Fields

In SDX Configuration Editor, you can modify the following field in the Proxy Router Driver section of the Router pane in an SAE configuration file.

SAE Community Manager

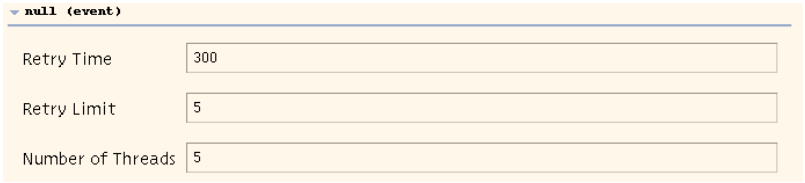
- Name of the community manager that manages network device communities. Active SAEs are selected from this community. You define community managers in the Ext. Interfaces tab of SDX Configuration Editor. See *Configuring the SAE Community Manager* on page 34.
- Value—Community name
- Default—PROXYCommunityManager
- Property name—Router.proxy.community.name

Configuring SAE Properties for the Event Notification API

To use SDX Configuration Editor to specify the name of the community manager that manages third-party network device communities:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.

2. Select the Ext. Interface tab, and expand the Proxy Router Driver section.



3. Edit or accept the default values in the field.

   See *Event API Fields* on page 37.

4. Select **File > Save**.

5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Event API Fields

In SDX Configuration Editor, you can modify the following fields in the Event API section of the Ext. Interface pane in an SAE configuration file.

Retry Time

- Amount of time between attempts to send events that could not be delivered.
- Value—Number of seconds in the range 0–2147483647
- Default—300
- Property name—SAEFeature.event.retry_time

Retry Limit

- Number of times an event fails to be delivered before the event is discarded.
- Value—Integer in the range 0–2147483647
- Default—5
- Property name—SAEFeature.event.retry_limit

Number of Threads

- Number of threads allocated to process events.
- Value—Integer in the range 0–2147483647
- Default—5
- Property name—SAEFeature.event.num_threads

Developing Initialization Scripts for Network Devices

When the SAE establishes a connection with a network device, it can run a script to customize the setup of the connection. These scripts are run when the connection between a network device and the SAE is established and again when the connection is dropped.

We provide the IorPublisher script in the */opt/UMC/sae/lib* folder. The IorPublisher script publishes the IOR of the SAE in the directory so that a NIC can associate a router with an SAE.

Interface Object Fields

Scripts for network devices interact with the SAE through an interface object called Ssp. The SAE exports a number of fields through the interface object to the script and expects the script to provide the entry point to the SAE.

Table 5 describes the fields that the SAE exports.

Table 5: Exported Fields

Ssp Attribute	Description
Ssp.properties	System properties object (class: java.util.Properties)—The properties should be treated as read-only by the script.
Ssp.errorLog	Error logger—Use the SsperrorLog.printIn (message) to send error messages to the log.
Ssp.infoLog	Info logger—Use the Ssp.infoLog.printIn (message) to send informational messages to the log.
Ssp.debugLog	Debug logger—Use the Ssp.debugLog.printIn (message) to send debug messages to the log.

The script must set the field Ssp.routerInit to a factory function that instantiates a router initialization object:

- < VRName > —Name of the virtual router object that has been configured for the network device in the format: virtualRouterName@RouterName
- < virtualIp > —Virtual IP address of the SAE (string, dotted decimal; for example: 192.168.254.1)
- < realIp > —Real IP address of the SAE (string, dotted decimal; for example, 192.168.1.20)
- < VRIp > —IP address of the virtual router (string, dotted decimal)
- < transportVR > —Name of the virtual router

The factory function must implement the following interface:

```
Ssp.routerInit(VRName,
virtualIp,
realIp,
VRIp,
transportVR)
```

The factory function returns an interface object that is used to set up and tear down a connection for a COPS server. A common case of a factory function is the constructor of a class.

The factory function is called directly after a connection is established. In case of problems, an exception should be raised that leads to the termination of the connection.

Required Methods

Instances of the interface object must implement the following methods:

- *setup()*—Is called when the connection is established and is operational. In case of problems, an exception should be raised that leads to the termination of the connection.
- *shutdown()*—Is called when the connection is terminated to the virtual router. This method should not raise any exceptions in case of problems.

Example: Router Initialization Script

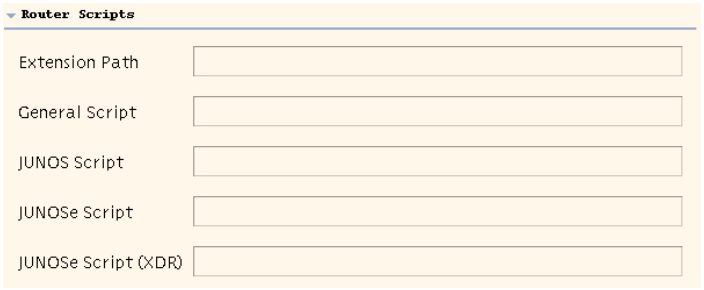
The following script defines a router initialization class named *SillyRouterInit*. The interface class does not implement any useful functionality. The interface class just writes messages to the infoLog when the router connection is created or terminated.

```
class SillyRouterInit:
    def __init__(self, vrName, virtualIp, realIp, vrIp, transportVr):
        """ initialize router initialization object """
        self.vrName = vrName
        Ssp.infoLog.printin("SillyRouterInit created")

    def setup(self):
        """ initialize connection to router """
        Ssp.infoLog.printin("Setup connection to VR %(vrName)s" %
                    vars(self))

    def shutdown(self):
        """ shutdown connection to router """
        Ssp.infoLog.printin("Shutdown connection to VR %(vrName)s" %
                    vars(self))

#
# publish interface object to Ssp core
#
Ssp.routerInit = SillyRouterInit
```

Specifying Router Initialization Scripts on the SAE

To use SDX Configuration Editor to specify router scripts:

1. In the navigation pane, select a directory configuration object for the SAE that you want to configure.
2. Select the Router tab, and expand the Router Scripts section.



3. Edit or accept the default values in the appropriate fields.

   See *Router Script Fields* on page 40.

4. Select **File > Save**.

5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Router Script Fields

In SDX Configuration Editor, you can edit the following fields in the Router Scripts section of the Router pane in an SAE configuration file.

Extension Path

- Path to router initialization scripts that are not in the default location, */opt/UMC/sae/lib*.
- Value—List of paths separated by semicolons (;)
- Default—No value
- Property name—Extension.path

General Script

- Router initialization script that can be used for all types of routers that the SRC software supports. The script is run when the connection between a router and the SAE is established and again when the connection is dropped.
- Value—Name of a script
- Default—No value
- Property name—Router.script.*

Using SNMP to Retrieve Information from Network Devices

You can use SNMP to retrieve information from a network device. For example, if you create a script that uses SNMP, you need to specify the SNMP communities that are on the network device.

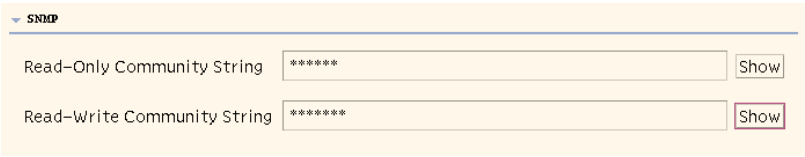
We recommend that you specify SNMP communities for each virtual router object. (See *Adding Virtual Router Objects* on page 28.) You can also configure global default SNMP communities.

Configuring Global SNMP Communities in the SRC Software

You can configure global default SNMP communities that are used if a VR does not exist on the router or the community strings have not been configured for the VR. To use SDX Configuration Editor to configure global default SNMP communities:

1. In the navigation pane, select a directory configuration object for the SAE that you want to configure.

2. Select the Router tab, and expand the SNMP section.



3. Edit or accept the default values in the fields.

   See *Global SNMP Community Fields* on page 41.

4. Select **File > Save**.

5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Global SNMP Community Fields

In SDX Configuration Editor, you can edit the following fields in the Router pane in an SAE configuration file.

Read-Only Community String

- Default SNMP community string used for read access to the router.
- Value—SNMP community string that matches a read-only community string configured on the router
- Default—Public
- Property name—Router.read-only.community.string

Read-Write Community String

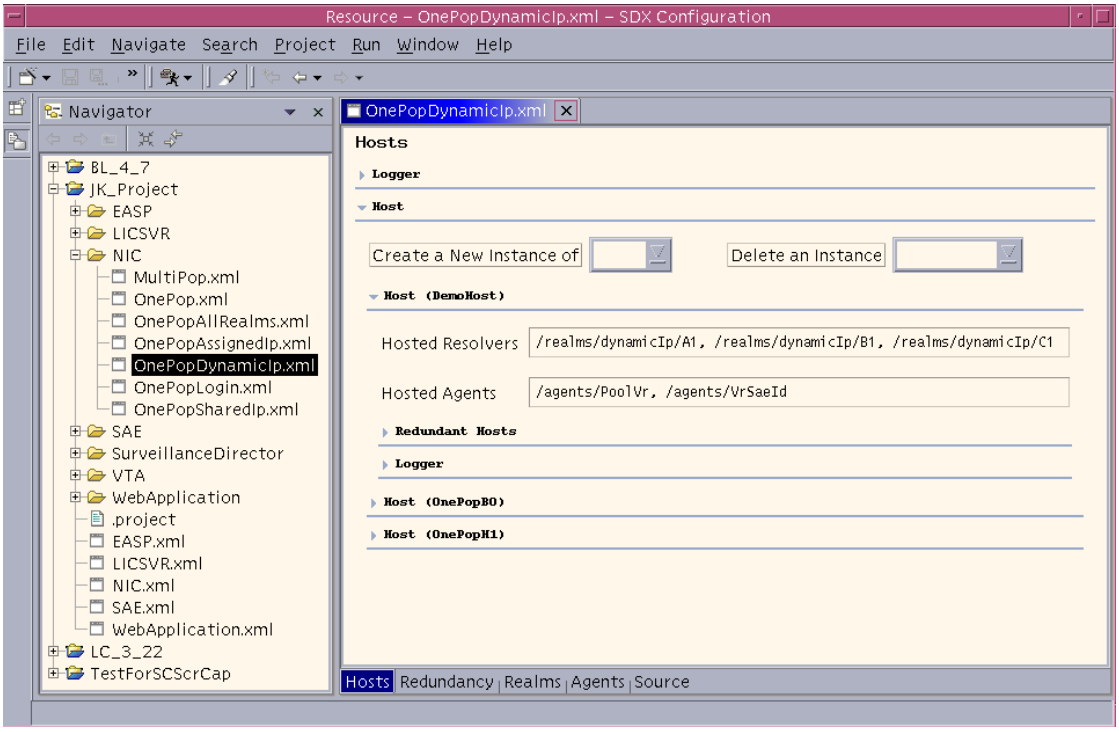
- Default SNMP community string used for write access to the router.
- Value—SNMP community string that matches a read-write community string configured on the router
- Default—Private
- Property name—Router.read-write.community.string

Using the NIC Resolver

If you are using the assigned IP subscriber method of logging in subscribers, and you are using the NIC to determine the subscriber's SAE, you need to configure a resolver on the NIC. The OnePopDynamicIp sample configuration data supports this scenario. The OnePopDynamicIp configuration supports one point of presence (POP) and provides no redundancy. The realm for this configuration accommodates the situation in which IP pools are configured locally on each virtual router object.

You can access the OnePopDynamicIp configuration in either SDX Admin or SDX Configuration Editor. Figure 7 shows the sample configuration in SDX Configuration Editor.

Figure 7: OnePopDynamicIP Sample Configuration in SDX Configuration Editor



For more information about the resolution process for OnePopDynamicIp, see *SRC-PE Network Guide: SAE, Juniper Networks Routers, and NIC.*