Chapter 20

Reviewing the NIC Configuration

In most cases, you use the network information collector (NIC) configuration scenarios supplied in the sample data and make minor changes to the configuration to enable the configuration scenario to work in your environment.

For information about modifying NIC configuration scenarios for your use, see *Chapter 9, Locating Subscriber Information with the NIC*.

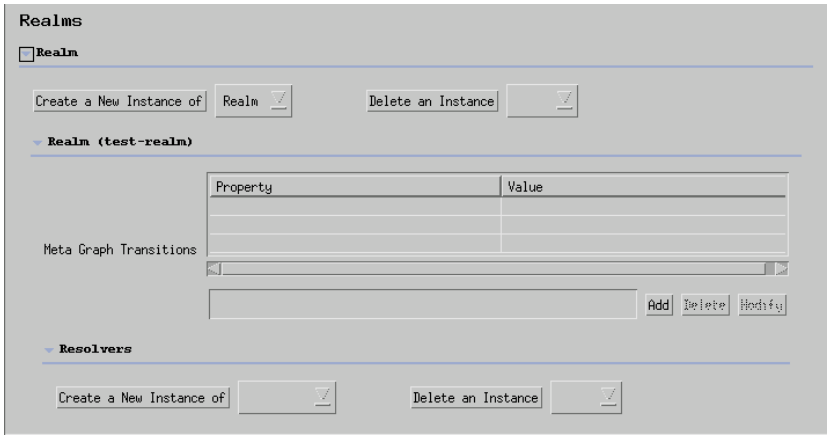
This chapter describes the NIC configuration as it appears in SDX Configuration Editor. Topics include:

Reviewing the Configuration for NIC Realms

NIC realms organize resolvers that perform a series of resolution tasks. Typically, you do not need to change the configuration for NIC realms.

To use SDX Configuration Editor to review the configuration for NIC realms:

1. In the navigation pane, select a configuration file for NIC.
2. Select the **Realms** tab, and expand the **Realm** section.



3. Review the transitions (sequential resolutions) in the Meta Graph Transitions section.

   The transition has the format property = value.

   If you want to qualify a property or value in a transition:

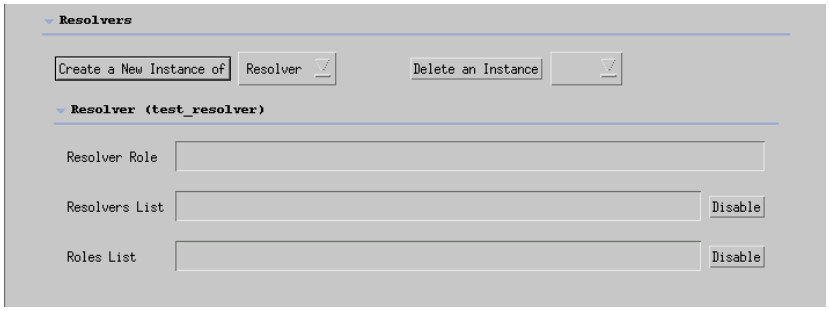
   a. Select a transition.

   b. Edit the entry.

   c. Click **Modify**.

   See *NIC Resolution Transition Fields* on page 299.

4. Expand the **Resolvers** section.



5. Review the configuration for resolvers to the realm.

   See *NIC Resolvers Fields* on page 300.

NIC Resolution Transition Fields

In SDX Configuration Editor, you can view the transitions (sequential resolutions) that occur in the resolution process in the Realms section of the Realms pane in a NIC configuration file. The following fields define the transitions.

Property

- Sequential number of the transition
- Value—Integer
- Example—2
- Property name— < number >

Value

- Transitions that define the resolution process.
- Value—List of transitions, each of which has the format < key > ':' < value > ':' < role > ':' ('[< 'constraint' >]' (,'[' < constraint > ']')*)* ':' '*'
  - ''—Literal value, for example ':' means a colon
  - < key > —Data type that is the key for the resolution process
  - < value > —Data type that is the value for the resolution process
  - < role > —Name of the resolver that handles this resolution process
  - ( )—Optional elements

- < constraint > —Mandatory or optional condition that must or may be satisfied before the next stage of the resolution process can proceed. For dynamic constraints, < constraint > is the data type for the constraint (for example: IpPool). For static constraints, < constraint > has the format < type > = < value > .
  - type—Data type for the constraint.
  - value—Value to check against. Must be in the proper format expected by the data type (for example: Domain = virneo.com).
  - Append a '?' to the constraint if the constraint is optional.
  - *—Operator that specifies that the resolver should accept the first response if it obtains information from multiple resolvers
- Example—2 = IpPool:Vr:L3:[IpPool][Domain?]

NIC Resolvers Fields

In SDX Configuration Editor, you can modify the following fields in the Resolvers section of the Realms pane in a NIC configuration file.

Resolver Role

- Role that this resolver executes.
- Value—Name of the role defined in the Meta Graph Transitions table at the top of the Realm section.
- Example—RoleA

Resolvers List

- Names of NIC resolvers to which this resolver forwards events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas

Roles List

- Names of NIC roles to which this resolver forwards events.
- Value—List of roles separated by commas
- Example—RoleA

Consolidator Agents

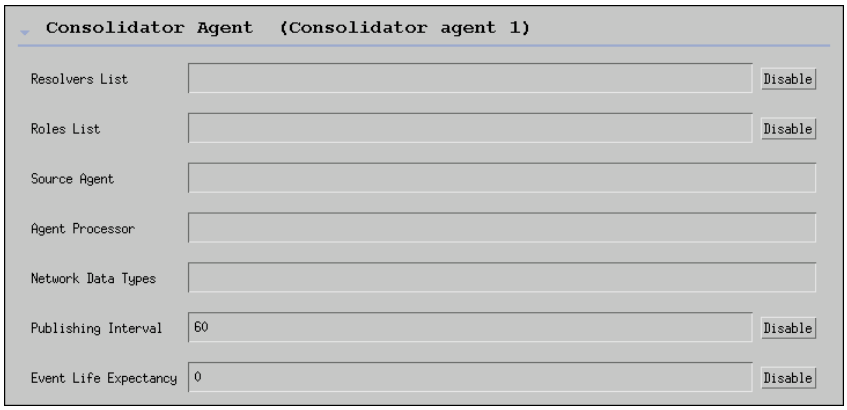
Consolidator agents are active NIC agents that publish data for passive agents, agents that provide information only on request. A consolidator agent comprises:

- A processor that takes a data mapping from a passive agent and returns either a network data object or a data mapping object that the consolidator agent then makes available.
- A component that tracks the number of times the processor adds or deletes an object. When the processor adds or deletes an object, this component publishes data for its associated resolvers.

Reviewing the Configuration of Consolidator Agents

To use SDX Configuration Editor to review the configuration for NIC consolidator agents:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Agents** tab, and expand the **Consolidator Agent** section.



3. Review the entries in the fields.

See *Consolidator Agent Fields* on page 301.

Consolidator Agent Fields

In SDX Configuration Editor, you can modify the following fields in the Consolidator Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas
- Default—No value
- Property name—pushToServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format <realmName>:<roleName>.
  - <realmName>—Name of realm
  - <roleName>—Name of role

- Default—No value
- Property name—pushToRole

Source Agent

- Path to the agent for which this consolidator agent publishes data.
- Value—Text string
- Default—No value
- Example—/agents/InterfaceIdInterface
- Property name—sourceAgent

Agent Processor

- Name of the Java class that the NIC agent uses to generate the data value object.
- Value—Path to Java class
- Default—net.juniper.smgt.gateway.nic.agent.dir.consolidator.RouterEventProcessor
- Property name—processor.classname

Network Data Types

- Data types that the agent publishes; for names of data types, see *Chapter 18, NIC Resolution Process*.
- Value— < key > or < key > , < value >
  - < key > —Name of data key
  - < value > —Name of data value
- Default—No value
- Example—IpPool, InterfaceId
- Property name—NetworkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
  - 0—Data does not expire
  - Other values—Actual life expectancy of data

- Default—0
- Property name—eventLifeExpectancy

Directory Agents

Directory agents obtain information from the directory and are active NIC agents; they provide information by:

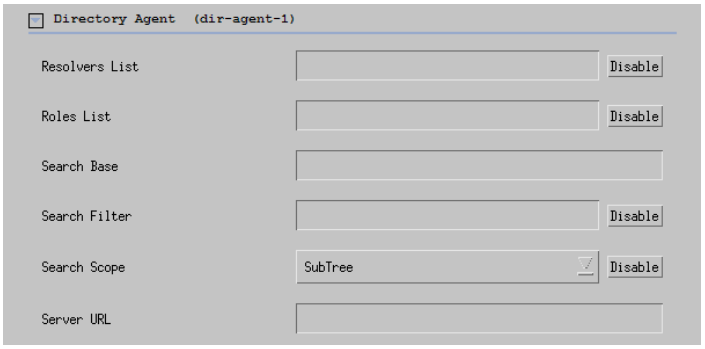
- Starting at a specified DN, reading directory entries that match the configured filter.
- Making available the directory entries that they read.
- Monitoring and publishing changes in the directory.

Reviewing the Configuration of Directory Agents

To use SDX Configuration Editor to review the configuration for NIC directory agents:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Agents** tab, and expand the **Directory Agent** section.

   The following sample Directory Agent section, shows a subset of the fields available for configuration.



3. Review the entries in the fields.

   See *Directory Agent Fields* on page 304.

Directory Agent Fields

In SDX Configuration Editor, you can modify the following fields in the Directory Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—Comma-separated list of paths to NIC resolvers relative to the Static Configuration object.
- Default—No value
- Example—/realms/ip/B1
- Property name—pushtoServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format <realmName>:<roleName>
  - <realmName>—Name of realm
  - <roleName>—Name of role
- Default—No value
- Property name—pushToRole

Search Base

- DN of the location in the directory from which the agent should read information.
- Value—<DN>, <base>
- Default—No value
- Default—*o=Network,* <*base*>
- Property name—baseDN

Search Filter (optional)

- Directory search filter that the agent should use.
- Value—LDAP search filter
- Default—No value
- Example—(objectclass=umcVirtualRouter)
- Property name—searchFilter

Search Scope (optional)

- Location in the directory relative to the base DN from which the NIC agent can retrieve information.
- Value—One of the following options:
  - Object—Entry specified in the Search Base field only
  - Level—Entry specified in the Search Base field and objects that are subordinate by one level
  - Subtree—Subtree of entry specified in the Search Base field
- Default—Subtree
- Property name—searchScope

Server URL

- Location of the directory in URL string format.
- Value—Location of the directory that stores configuration information in URL string format ldap:// < host > : < portNumber >
  - < host > —IP address or name of directory host
  - < portNumber > —Number of TCP/IP port
- Default—No value
- Example—ldap://127.0.0.1:389/
- Property name—java.naming.provider.url

Backup Servers URL

- List of redundant directories.
- Value—List of URLs separated by semicolons
- Default—No value
- Example—ldap://127.0.0.1:389/
- Property name—net.juniper.smgt.des.backup_provider_urls

Authentication DN

- DN that contains the username that the directory server uses to authenticate the NIC agent.
- Value— < DN > , < base >
- Default—No value
- Example—*cn = nic, ou = Components, o = Operators,* < *base* >
- Property name—java.naming.security.principal

Password

- Password that the directory server uses to authenticate the NIC agent.
- Value— < password >
- Guidelines—The password can be encoded in base64 and not visible in plain text. To use an encoded value, use the format {BASE64} < encoded-value > .

- Default—No value
- Example—nic
- Property name—java.naming.security.credentials

Key Attribute Name(s)

- Name of the directory attribute that the NIC agent uses for the network data object called *key*.
- Value—Name of an attribute in the directory
- Default—No value
- Example—virtualRouterName
- Property name—key.attrNames

Key Attribute Processor

- Java class that the NIC agent uses to generate the data key object.
- Value—Path to Java class
- Default—No value
- Example—net.juniper.smgt.gateway.nic.agent.dir.DnAttributeProcessor
- Property name—key.processor.classname

Value Attribute Name(s)

- Directory attribute that the NIC agent uses for the network data object called *value*.
- Value—Name of an attribute in the directory
- Guidelines—Specify only if the agent publishes mappings.
- Default—No value
- Example—SaeId
- Property name—value.attrNames

Value Attribute Processor

- Name of the Java class that the NIC agent uses to generate the data value object.
- Value—Path to Java class
- Guidelines—Specify only if the agent publishes mappings.
- Default—No value
- Property name—net.juniper.smgt.gateway.nic.agent.dir.vr.VrAttributeProcessor

Network Data Types

- Names of the data types that this NIC agent publishes. For names of data types, see *Chapter 18, NIC Resolution Process*.
- Default—No value

- Example—IpPool,Vr
- Property name—networkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Example—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
  - 0—Data does not expire
  - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Directory Eventing Fields

For information about the directory eventing fields, see *SRC-PE Getting Started Guide, Chapter 32, Distributing Directory Changes to SRC Components on a Solaris Platform*.

SAE Plug-In Agents

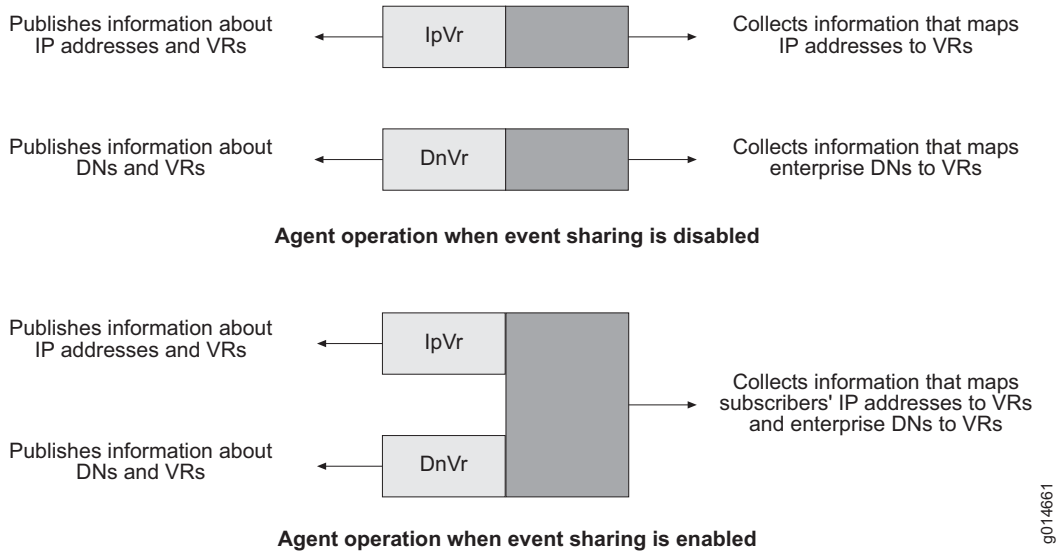
SAE plug-in agents collect information about the subscribers and interfaces managed by an SAE. For example, an SAE plug-in agent can collect and make available mappings of subscribers' IP addresses to the VRs from which those subscribers connect to the network. An SAE plug-in agent can also obtain information about the managed interfaces on a router.

We recommend that you have state synchronization enabled for SAE plug-in agents and that you use NIC replication to maintain high availability for the NIC. SAE plug-in agents always have information that is current unless the agent is overloaded and has not processed some of the SAE events; then the unprocessed events are placed in a queue.

You do not need to configure redundancy for the SAE plug-in agent as in past releases. If you have an SAE plug-in agent that uses agent redundancy, we recommend that you enable state synchronization for the agent and use NIC replication.

Each SAE plug-in agent is composed of two parts: one part that publishes NIC events and another part that collects SAE events. You can use multiple SAE plug-in agents supported on a single host to share the part that collects events. In this case, the SAE regards the agents as one plug-in, although the NIC still regards the agents as separate entities. Figure 12 illustrates this concept.

Figure 12: Event Sharing for SAE Plug-In Agents

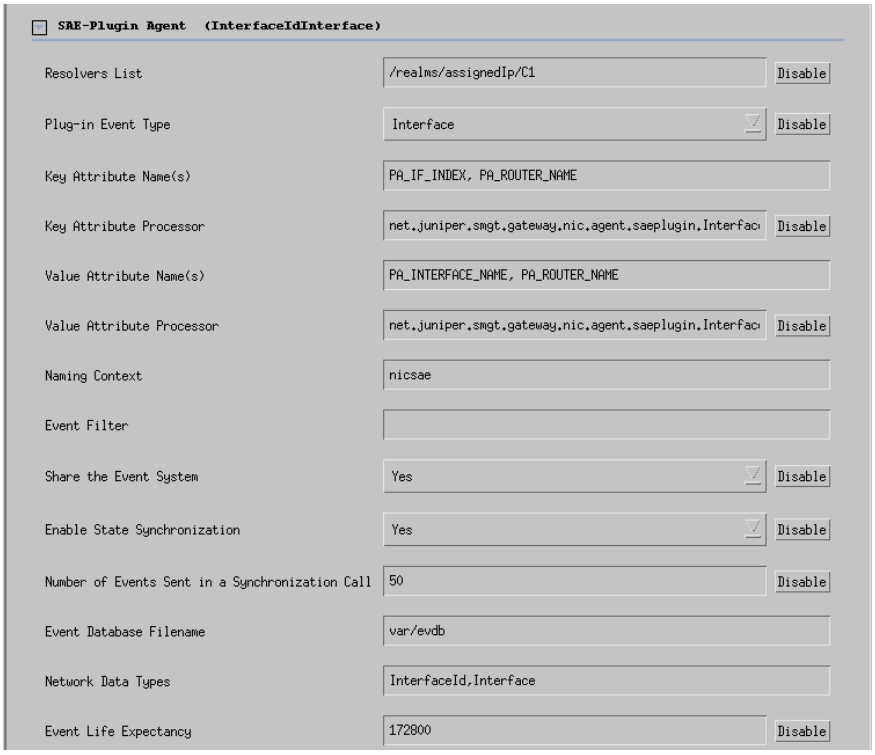


For more information about SAE plug-ins, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 8, Overview of Plug-Ins Included with the SAE*.

Reviewing the Configuration of SAE Plug-In Agents

To use SDX Configuration Editor to review the configuration for NIC SAE plug-in agents:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Agents** tab, and expand the **SAE Plug-In Agent** section.



3. Review the entries in the fields.

See *SAE Plug-In Agent Fields* on page 309.

SAE Plug-In Agent Fields

In SDX Configuration Editor, you can modify the following fields in the SAE Plug-In Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—Comma-separated list of paths to NIC resolvers relative to the Static Configuration object.
- Default—No value
- Example—/realms/dn/B1
- Property name—pushToServer

Plug-in Event Type

- Types of plug-in events that the agent supports.
- Value
  - User—Agent supports user-tracking plug-in events.
  - Interface—Agent supports interface-tracking plug-in events.
- Default—User
- Property name—pluginEventType

Key Attribute Name(s)

- List of plug-in attributes that provide information for the data key.
- Value—List of comma-separated plug-in attributes
- Guidelines—The list can contain one or more plug-in attributes.

  If the format of the single plug-in attribute is not a string or you specify multiple plug-in attributes, the agent passes the data to the key processor to construct the data value in string format. In this case, you must specify the processor in the Key Attribute Processor field.

- Default—No value
- Example—PA_USER_DN, PA_ROUTER_NAME
- Property name—key.pluginAttributeNames

Key Attribute Processor

- Name of the Java class that the agent uses to generate the data key object.
- Value—Path to Java class
- Guidelines—Configure a key attribute processor if the agent acquires for the key value either a single plug-in attribute that is not in string format or multiple plug-in attributes.
- Default—No value
- Example—net.juniper.smgt.gateway.nic.agent.saeplugin.InterfaceIdProcessor
- Property name—key.processor.classname

Value Attribute Name(s)

- List of plug-in attributes that provide information for the data value.
- Value—List of comma-separated plug-in attributes
- Guidelines—The list can contain one or more plug-in attributes.

  If the format of the single plug-in attribute is not a string or you specify multiple plug-in attributes, the agent passes the data to the value processor to construct the data value in string format. In this case, you must specify the processor in the Value Attribute Processor field.

- Default—No value
- Example—PA_USER_DN, PA_ROUTER_NAME
- Property name—value.pluginAttributeNames

Value Attribute Processor

- Name of the Java class that the agent uses to generate the data value object.
- Value—Path to Java class
- Guidelines—Configure a value attribute processor if the agent acquires for the data value either a single plug-in attribute that is not in string format or multiple plug-in attributes.
- Default—No value
- Example—net.juniper.smgt.gateway.nic.agent.saeplugin.InterfaceProcessor
- Property name—value.processor.classname

Naming Context

- CORBA naming context in which the agent publishes references.
- Value—String that must match the context name in the objectref property for this SAE plug-in

  See *Chapter 18, NIC Resolution Process*.

- Guidelines—If you configure event sharing for multiple SAE plug-in agents, this setting must be identical for all those agents.
- Default—No value
- Example—nicsaetestDNOttawa

  This example matches the context name of the following objectref property: corbaname::10.10.10.10:900/NameService#nicsaetestDNOttawa/saePort

  - 10.10.10.10—Address of the machine running the CORBA naming server
  - 900—TCP/IP port
  - saePort—Name of plug-in (in this case, the agent eventing system)
- Property name—pluginNamingCtx

Event Filter

- LDAP filter that restricts the events that the agent collects.
- Value— < pluginAttribute > = < attributeValue >
  - < pluginAttribute > —Plug-in attribute name
  - < attributeValue > —Value of filter
- Default—No value
- Example—PA_USER_TYPE = INTF
- Property name—eventFilter

Share the Event System

- Specifies whether or not the agent shares the event system with other agents in the same host.
- Value
  - Yes—Agent shares the event system.
  - No—Agent does not share the event system.
- Guidelines—If you configure event sharing for multiple SAE plug-in agents, this setting must be identical for all those agents.
- Default—No value
- Property name—eventSharingEnabled

Enable State Synchronization

- Specifies whether or not the state of the agent can be synchronized from the SAE. With state synchronization enabled, the state of the agent can be synchronized at any time.
- Value—Yes or No
- Guidelines—With state synchronization enabled, the agent uses NIC host redundancy.
- Default—Yes
- Property name—stateSyncEnabled

Number of Events Sent in a Synchronization Call

- Number of events the SAE sends to the agent at one time during state synchronization.
- Value—Integer in the range 1—2147483647
- Guidelines—This value is used if Enable State Synchronization is set to Yes.
- Default—50
- Property name—stateSyncBulkSize

Event Database Filename

- File in which the agent stores event information.
- Value—Path, relative to the directory that contains the NIC software, to the file
- Guidelines—If you configure event sharing for multiple SAE plug-in agents, this setting must be identical for all those agents.
- Default—No value
- Example—var/evdbDNMontral
- Property name—eventDbFilename

Network Data Types

- Attribute names for data that the agent collects in the format <key>, <value>
- Value
  - <key>—Attribute name for the data key
  - <value>—Attribute name for the data value
- Default—No value
- Example—Dn, Vr
- Property name—networkDataTypes

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
  - 0—Data does not expire
  - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Properties Agents

A properties agent retrieves information from one or more specified property files and makes event information based on the information in the file available to the NIC. The format of the property file must comply with the typical format used for Java properties.

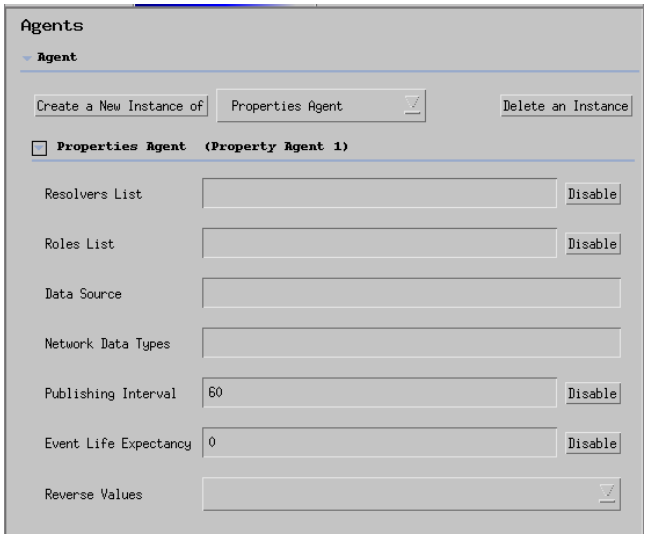
By default, property names in the file are keys, and property values are key values. If a property name (key) appears more than once in a file, the NIC uses the last value for the key in the file.

Although a properties agent may be used by an SRC application, typically you do not need to configure it. If you want to use a properties agent with a configuration scenario, contact Juniper Networks Professional Services.

Configuring Properties Agents

To use SDX Configuration Editor to configure properties agents:

1. In the navigation pane, select a NIC configuration file.
2. Select the **Agents** tab, and expand the **Properties Agent** section.



3. Fill in the entries in the fields.

   See *Properties Agent Fields* on page 314.

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

Properties Agent Fields

In SDX Configuration Editor, you can modify the following fields in the Properties Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas
- Default—No value
- Example— /realms/staticRouteIp/C1
- Property name—pushToServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format <realmName>:<roleName>.
  - <realmName>—Name of realm
  - <roleName>—Name of role
- Default—No value
- Property name—pushToRole

Data Source

- List of URIs of property files that provides information about NIC events to the NIC system.
- Value—URIs separated by commas
- Guidelines—You must provide at least one URI.
- Default—No value
- Property name—dataSource

Network Data Types

- Attribute names for data that the agent collects in the format <key>, <value>
- Value
  - <key>—Attribute name for the data key
  - <value>—Attribute name for the data value
- Default—No value
- Example—Dn, Vr
- Property name—networkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
  - 0—Data does not expire
  - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Reverse Values

- Specifies whether a property name is made available as a NIC key or a NIC value
- Value
  - Yes—Property names made available as keys
  - No—Property names made available as values
- Default—No value
- Property name—reverseValues

XML Agents

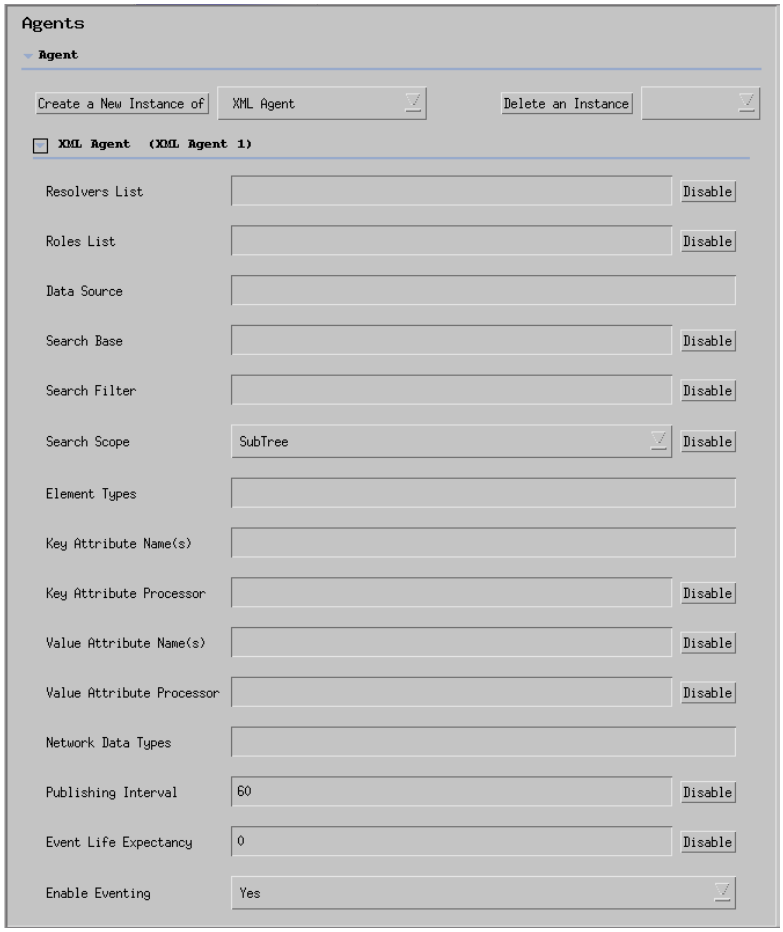
An XML agent retrieves information from a specified XML document and makes information available to NIC based on specified tags in the file. An XML agent provides information about one type of data or mappings.

Although an XML agent may be used by an SRC application, typically you do not need to configure it. If you want to use an XML agent with a configuration scenario, contact Juniper Networks Professional Services.

Configuring XML Agents

To use SDX Configuration Editor to configure XML agents:

1. In the navigation pane, select a NIC configuration file.
2. Select the **Agents** tab, and expand the **XML Agent** section.



3. Modify the entries in the fields.

   See *XML Agent Fields* on page 317.

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

XML Agent Fields

In SDX Configuration Editor, you can modify the following fields in the XML Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas
- Default—No value
- Example— /realms/staticRouteIp/C1
- Property name—pushToServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format <realmName>:<roleName>
  - <realmName>—Name of realm
  - <roleName>—Name of role
- Default—No value
- Property name—pushToRole

Data Source

- URI of the XML document that provides information about NIC events to the NIC system.
- Value—<URI>
- Guidelines—You must provide a URI for the XML document.
- Default—No value
- Property name—dataSource

Search Base

- Root XML element in the specified XML document at which the agent starts to search the XML document.
- Value—<XML element>
- If you do not specify an element for the search base, the agent starts searching at the top of the file.
- Default—No value
- Property name—base

Search Filter

- Search filter the agent uses to parse an XML document.
- Value—Search filter syntax defined in RFC 2254—The String Representation of LDAP Search Filters (December 1997)
- Default—No value

Search Scope

- Level at which the agent searches the XML document.
- Value
  - Object—Searches the object defined by the search base entry.
  - One level—Specifies objects at the same level as the object defined by the search base entry.
  - Subtree—Searches objects subordinate to object defined by the search base entry.
- Default—No value

Element Types

- Types of XML elements that the agent will use.
- Value—List of XML elements, separated by commas.
- Guidelines—Elements in this list must contain the attributes that the agent makes available.
- Default—No value
- Property name—elementTypes

Key Attribute Name(s)

- Name of the directory attribute that the NIC agent uses for the network data object called *key*.
- Value—Name of an attribute in the directory
- Default—No value
- Example—virtualRouterName
- Property name—key.attrNames

Key Attribute Processor

- Java class that the NIC agent uses to generate the data key object.
- Value—Path to Java class
- Default—No value
- Example—net.juniper.smgt.gateway.nic.agent.dir.DnAttributeProcessor
- Property name—key.processor.classname

Value Attribute Name(s)

- Directory attribute that the NIC agent uses for the network data object called *value*.
- Value—Name of an attribute in the directory
- Guidelines—Specify only if the agent publishes mappings.
- Default—No value
- Example—SaeId
- Property name—value.attrNames

Value Attribute Processor

- Name of the Java class that the agent uses to generate the data value object.
- Value—Path to Java class
- Default—No value
- Property name—value.processor.classname

Network Data Types

- Attribute names for data that the agent collects in the format <key>, <value>
- Value
  - <key>—Attribute name for the data key
  - <value>—Attribute name for the data value
- Default—No value
- Example—Dn, Vr
- Property name—networkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
  - 0—Data does not expire
  - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Enable Eventing

- Specifies whether or not the agent monitors changes to the XML document and sends events when changes occur.
- Value
  - Yes—Agent sends events.
  - No—Agent does not send events.
- Default—Yes
- Property name—enableEvents

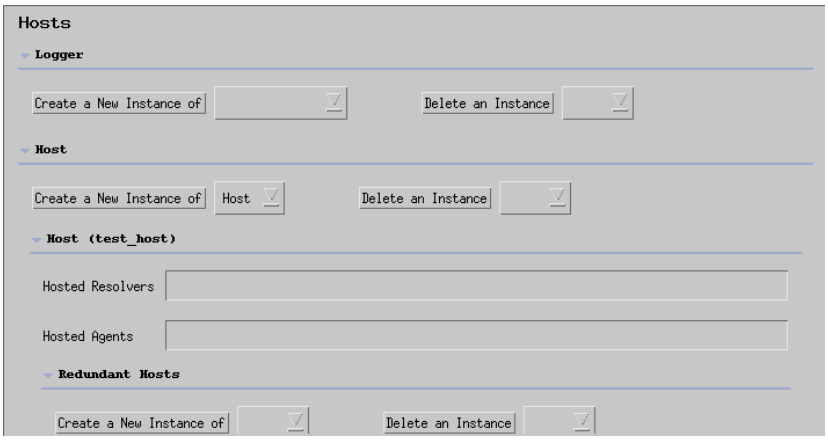
Reviewing and Changing the Configuration for a NIC Host Instance

Typically, you use the DemoHost NIC host in the sample data. You review the configuration for the NIC host associated with a scenario on a system from SDX Configuration Editor.

To review the configuration for an instance of a NIC host:

1. Click the **Hosts** tab in a NIC configuration file.
2. Expand the Host entry.

   The host details appear in the Hosts pane.



3. Review information about the NIC components that this host supports.

   See *NIC Host Fields* on page 322.

NIC Host Fields

in SDX Configuration Editor, you can modify the following fields in the Host section of the Hosts pane in a NIC configuration file.

Hosted Resolvers

- Names of NIC resolvers that this host manages.
- Value—Comma-separated list of paths to NIC resolvers
  - Paths show the locations of the NIC resolvers relative to the Static Configuration object.
  - Subfolders in a path are separated by the forward slash (/).
- Example—/realms/sharedIp/A1, /realmssharedIp/B1, /realms/sharedIp/C1, /realms/dn/A1, /realms/dn/B1, /realms/dn/C1
- Property name—Server

Hosted Agents

- List of paths to NIC agents that this host supports.
- Value—Comma-separated list of paths to agents
  - Paths show the locations of the NIC agents relative to the Static Configuration object.
  - Subfolders in a path are separated by the forward slash (/).
- Example—/agents/VrSaeId, /agents/Router, /agents/PoolInterfaceId, /agents/InterfaceIdInterface
- Property name—Agent

Configuring Logging for NIC

You can configure one set of logging properties for all NIC hosts associated with a configuration. You can also configure a set of logging properties for each individual NIC host.

To use SDX Configuration Editor to configure logging for NIC:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Hosts** tab.
3. To configure logging for individual NIC hosts, expand the **Logger** section subordinate to a host in the Hosts pane, and configure the fields.
4. To configure logging for all NIC hosts, expand **Logger** at the top of the Hosts pane, and configure the fields.

For information about logging properties and about managing log files, see:

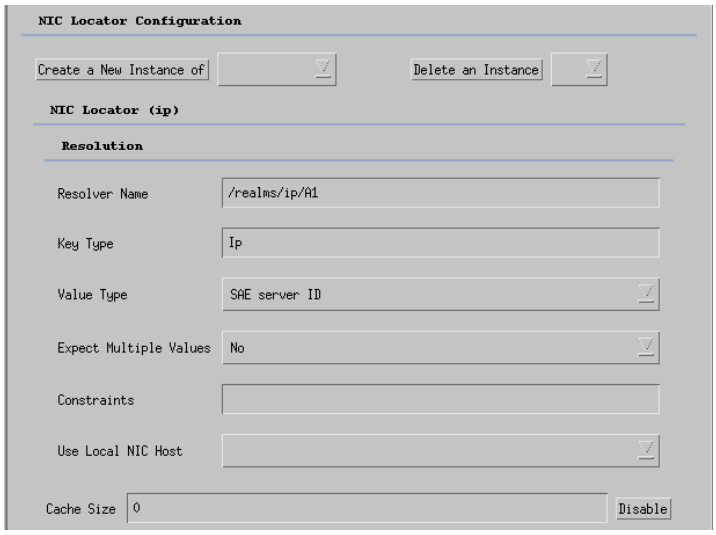
- *SRC-PE Monitoring and Troubleshooting Guide, Chapter 3, Configuring Logging for SRC Components with the CLI*
- *SRC-PE Monitoring and Troubleshooting Guide, Chapter 4, Configuring Logging for SRC Components on a Solaris Platform*

Reviewing the Configuration for NIC Locators

Each NIC configuration scenario includes configuration for NIC locators for each type of resolution in the scenario. Non-Java applications communicate with NIC locators through the NIC access interface to perform data resolutions. Typically, you do not need to change the configuration for NIC locators. The configuration fields for NIC locators are the same as the configuration fields for NIC proxies.

To use SDX Configuration Editor to review the configuration for NIC locators:

1. Set the editing level for SDX Configuration Editor to Normal, Advanced, or Expert.
2. In the navigation pane, select a NIC configuration file.
3. Select the **NIC Locators** tab; expand the **NIC Locator Configuration** section, then the **NIC Locator** section, and then the **Resolution** section.



4. Review the entries in the fields.

   The configuration fields for a NIC locator are the same as a subset of the fields for a NIC proxy.

   For information about NIC locator configuration fields, see the description of the NIC proxy fields in *Chapter 13, Configuring Applications to Communicate with an SAE*.

Managing NIC Resolvers

You can also view data mappings for NIC resolvers with a program called **nicDump**, which is installed in the folder */opt/UMC/nic/bin* when you install a NIC host on a machine.

When you run **nicDump**, you can view information for all resolvers that the NIC host supports or information about a specific NIC resolver. The program writes the output to the file you specify or to the default file, */opt/UMC/nic/var/dumpResult.txt*. The syntax for the command is:

nicDump [[[-r < resolverName >] [-f < fileName >]]| --help]

- < resolverName > —Name of the resolver for which you want to view information in the specified file; the value is case sensitive and must match the name in the NIC configuration.
- < fileName > —Name of the file in which you want to place state information for the specified resolver; the value is case sensitive.

To generate information about NIC resolvers with nicDump, run the program.

Example 1

```
# /opt/UMC/nic/bin/nicDump -r /realms/ip/A1 -f /opt/UMC/nic/A1_data.txt
# more /opt/UMC/nic/A1_data.txt

###### DUMP OUTPUT #####  Wed Sep 24 11:01:00 EDT 2003
Resolver = /realms/ip/A1
[IpPool:([10.227.25.208 10.227.25.255]), IpPool:([10.227.1.1
10.227.255.255]), IpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.
255])]
```

Example 2

```
# /opt/UMC/nic/bin/nicDump
# more /opt/UMC/nic/var/dumpResult.txt

###### DUMP OUTPUT #####  Tue Sep 23 13:22:32 EDT 2003
[IpPool:([10.227.25.208 10.227.25.255]), IpPool:([10.227.1.1
10.227.255.255]), IpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255])]
[pair<IpPool:([10.227.25.208 10.227.25.255]), Vr:default@phoenix>,
pair<IpPool:([10.227.1.1 10.227.255.255]), Vr:default@bigfoot>,
pair<IpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255]), Vr:default@erx-node1>]
```

[pair<Vr:default@erx-node1, SaeId:IOR:12345>, pair<Vr:default@bigfoot,
SaeId:IOR:8888888888>]

DUMP OUTPUT ##### Tue Sep 23 13:58:48 EDT 2003
[IpPool:([10.227.25.208 10.227.25.255]), IpPool:([10.227.1.1
10.227.255.255]), IpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255])]
[pair<IpPool:([10.227.25.208 10.227.25.255]), Vr:default@phoenix>,
pair<IpPool:([10.227.1.1 10.227.255.255]), Vr:default@bigfoot>,
pair<IpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255]), Vr:default@erx-node1>]
[pair<Vr:default@erx-node1, SaeId:IOR:12345>, pair<Vr:default@bigfoot,
SaeId:IOR:8888888888>]