Chapter 5

Configuring Subscriber–Related Properties on the SAE on a Solaris Platform

This chapter describes how to configure subscriber-related properties on the SAE by modifying a property file. You can use property files on a Solaris platform.

- To use the SRC CLI to configure an SAE on a Solaris platform. See *Chapter 4, Configuring Subscriber–Related Properties on the SAE with the SRC CLI*.
- To use the C-Web interface, see see *SRC-PE C-Web Interface Configuration Guide, Chapter 26, Configuring Subscriber–Related Properties on the SAE with the C-Web Interface*.

Topics in this chapter include:

Overview

The SAE property file contains SAE configuration data that is stored in the directory.

You can modify the SAE property file with SDX Admin, or a standard text editor. Each field description includes a property name, which is used if you modify the properties with SDX Admin or a text editor.

Modifying the SAE Property File

This section shows how to edit the property file with SDX Admin or a standard text editor.

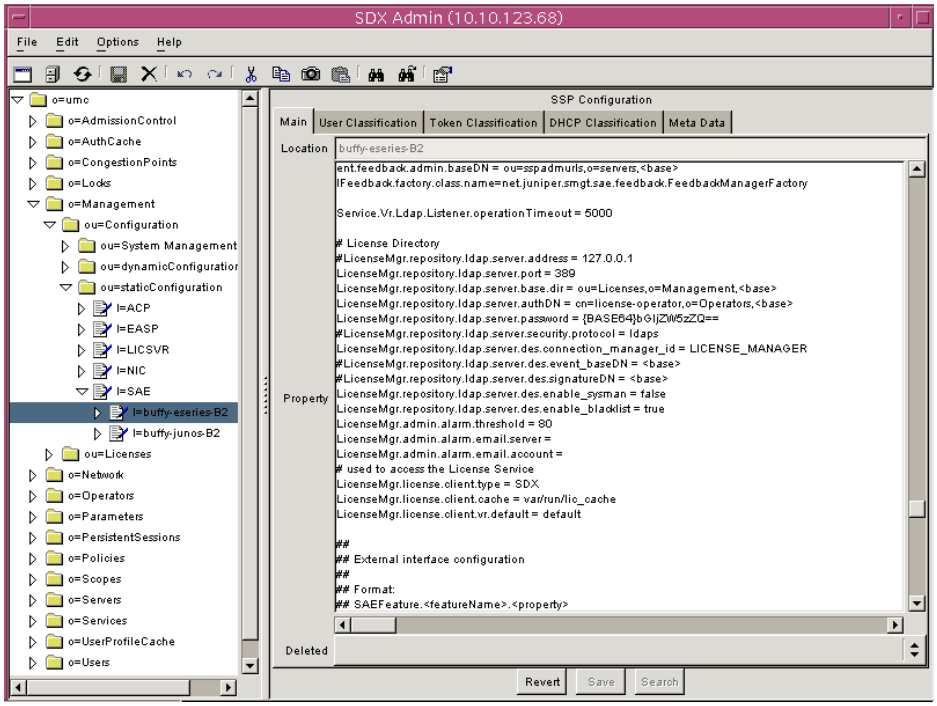
Editing Properties with SDX Admin

To edit the properties with SDX Admin:

1. In the SDX Admin navigation pane, access the object *I = SAE, ou = staticConfiguration, ou = configuration, o = management, o = umc*.

2. In this folder, click on the I = POP-ID object associated with this SAE.

   The SAE configuration appears in the Main tab in the SSP Configuration pane.



3. Scroll to the text you want to edit, or click **Search** to find an item in the configuration file.

4. Add or modify the relevant information, and click **Save**.

Editing Properties with a Text Editor

To edit the properties with a text editor:

1. Open a shell in the directory in which you installed the SAE.

   The default installation directory for SAE is */opt/UMC/sae*.

2. Download the properties to a file with the SAE configuration utility.

   etc/config -g <filename>

   A file called < filename >, which contains the SAE properties, appears in the *sae* subdirectory.

3. Edit the file with a text editor, such as VI or EMACS.

4. Update the object's file properties with the SAE configuration utility.

   etc/config -p <filename>

Loading Subscriptions Based on RADIUS Authorization

You can set up the SAE to load subscriptions based on values that it receives in RADIUS authorization response packets. For this method of loading subscriptions to work, the subscriber must be subscribed to the service.

To use this feature, you set up the RADIUS authorization plug-in to return the setLoadServices attribute, and you specify a regular expression in the SAE property file. When the plug-in returns the setLoadServices attribute, the SAE applies the regular expression to the string in the setLoadServices attribute.

There are two SAE properties that you can use to set the expression:

- SubscriptionParser.regex—Specifies the regular expression that is used to match a single service name.
- SubscriptionParser.auto—Specifies the number of a group of services that corresponds to activate-on-login services. That is, if a subscription is matched by this group, it is activated.

For example:

```
SubscriptionParser.regex = ([^;!]*);|([^;!]*)!
SubscriptionParser.auto = 2
```

A group match corresponds to a regular expression that is enclosed in (). In this example, the regular expression in the subscription parser contains two groups:

1. A string of characters other than ";" and "!", followed by ";"
2. A string of characters other than ";" and "!", followed by "!"

The value of 2 in the SubscriptionParser.auto property causes the second group of services—services followed by !—to be activated on login. For example, if the setLoadServices string is video-gold;audio-gold!, it is parsed to video-gold and audio-gold. The audio-gold subscription is activated provided that the subscriber is subscribed to audio-gold services.

NOTE: Persistent service sessions are not parsed. That is, if a subscriber has activated persistent service sessions, then these sessions are activated independent of the RADIUS authorization responses.

Another way to load subscriber services based on RADIUS authorization is to use the serviceBundle vendor-specific attribute (VSA) as input to the subscriber classification script and load different subscriber profiles based on the RADIUS response. Different subscriber profiles subscribe to different services. This approach gives wholesalers a basic tool to outsource service subscriptions to a retailer. The wholesaler and retailer must agree on a RADIUS attribute (for example, serviceBundle) that is provided by the retailer and interpreted by the SAE (that is, the wholesaler).

The subscription parser properties are located in */opt/UMC/sae/etc/dir.template*. (See *Modifying the SAE Property File* on page 49.)

SubscriptionParser.regex

- Regular expression that is applied to the setLoadServices attribute in RADIUS authorization response packets. The regular expression matches a single service name and is applied repeatedly until no match is found.
- Value—Regular expression; you can group matches by enclosing them in parenthesis ().
- Default—"([^;!]*);|([^;!]*)!"
- Property name—SubscriptionParser.regex

SubscriptionParser.auto

- Expression that identifies the number of a group of services that are to be activated automatically. If a subscription is matched by this group, it is automatically activated.
- Value—Expression
- Default—2
- Example—The default regular expression corresponds to a string of service names that are separated by "," or "!". If a service name is followed by "!", it is activated automatically.
- Property name—SubscriptionParser.auto

Accepting Login Names with Different Formats

You can configure the SAE to accept login names of different formats. For example, the format subscriberName@domainName is a common format for the login name of subscribers who connect through PPP; however, other subscribers may use other formats, such as domainName/userName.

To configure the SAE to accept these different formats, you specify a set of properties that parse the login name to obtain the userName and domainName objects for the subscriber. Each property contains a regular expression that includes one or two subexpressions—independent expressions in the complete regular expression—each of which is enclosed in parentheses.

The property for login name parsing has the form:

```
LoginName.parser.<number>.<userGroup>[.<domainGroup>] = \
<regular expression>
```

number

- Number that specifies the order in which the SAE should apply the property when it parses the loginName. The SAE applies the properties in the specified order from lowest to highest.

userGroup

- Number of the backreference that extracts the username.
- In the following example, the userGroup backreference is set to 1. This means that the first backreference in the expression ([^@]*) identifies the username:

  ```
  LoginName.parser.1.1.2 = ([^@]*)@(.*)
  ```

domainGroup

- Optional number of the backreference that extracts the domain name.
- In the following example, the domainGroup backreference is set to 1. Therefore, the first backreference in the expression ([^/]*) identifies the domain name:

  ```
  LoginName.parser.2.2.1 = ([^/]*)/(.*)
  ```

regular expression

- Regular expression that includes one or two subexpressions—independent expressions in the complete regular expression—each of which is enclosed in parentheses.
- When you define regular expressions for a domain name parser, you must include four backslashes (\\\\) to effect a single backslash. For example, suppose you define the following parser:

  ```
  LoginName.parser.1.2.1 = (.*?)[\\\\](.*?)
  ```

  This example parses the login name isp1\jane as:

  ```
  domain name: isp1
  username: jane
  ```

- For more information about using regular expressions for this feature, see: http://jakarta.apache.org/regexp/apidocs/org/apache/regexp/RE.html

Default Login Parser Properties

Table 10 shows default properties that the SAE uses to parse login names. Table 11 shows some examples of subscriber and domain names obtained through the default parsing properties.

Table 10: Default SAE Properties That Parse Login Names

Property	Function	Values
LoginName.parser.1.1.2 = ([^@]*)@(.*)	Parses login names of the format userName@domainName	LoginName.parser.1.1.2—First parser applied by the SAE to login names; first backreference identifies the username, and second backreference identifies the domain name. ([^@]*)—First backreference: username is a string of characters other than the at-sign (@). @—An at-sign precedes the domain name. (.*)—Second backreference: domain name is a string of characters.
LoginName.parser.2.2.1 = ([^/]*)/(.*)	Parses login names of the format domainName/userName	LoginName.parser.2.2.1—Second parser applied by the SAE to login names; second backreference identifies the subscriber name, and first backreference identifies the domain name. ([^/]*)—First backreference: domain name is a string of characters other than the forward slash (/). /—A forward slash precedes the username. (.*)—Second backreference: username is a string of characters.
LoginName.parser.3.1 = (.*)	Parses login names that contain no domain name	LoginName.parser.3.1—Third parser applied by the SAE to login names; first backreference identifies the username, no domain name. (.*)—First backreference: username is a string of characters.

Table 11: Examples of Subscriber and Domain Names Obtained from Default Properties

Login Name	Output from Default Parsing Properties
joeUser@isp1.com	■ The username is joeUser. ■ The domain name is isp1.com.
isp1/joe	■ The username is joe. ■ The domain name is isp1.
isp1/joe@isp2	■ The username is isp1/joe. ■ The domain name is isp2.