Chapter 7

# Classifying Interfaces and Subscribers on a Solaris Platform

This chapter provides information for configuring and using classification scripts with SDX Admin.

You can also use the SRC CLI to configure classification scripts on the C-series platform or on a Solaris platform. See *Chapter 9, Configuring Internal, External, and Synchronization Plug-Ins with the SRC CLI*.

Topics in this chapter include:



## Overview of Classification Scripts

The SAE uses classification scripts to determine whether it manages router interfaces, to select default policies, to find subscriber profiles, and to choose DHCP profiles. The SAE has three classification scripts:

- Interface classification script—When a subscriber's IP interface comes up on the router, the router sends the subscriber's login and interface information to the SAE. The SAE runs the interface classification script to determine whether the SAE manages the interface and if so, what default policies to send to the router.

- Subscriber classification script—If the SAE is managing the interface, the SAE uses the login and interface information that the router sends to run the subscriber classification script to determine which subscriber session to load into memory.

- DHCP classification script—For DHCP subscribers, the SAE runs DHCP classification scripts to choose DHCP profiles.

### How Classification Scripts Work

Classification scripts consist of *targets* and *criteria*.

- A target is the result of the classification script. For example, the result of subscriber classification scripts is an LDAP search string that is used to find a unique subscriber profile in the directory. The result of interface classification scripts is a policy group in the directory.

- Criteria are match criteria. The script attempts to match criteria in the script to information sent from the router. For example, match criteria for a subscriber classification script might be login type or domain name. Match criteria for an interface classification script could be interface IP address or interface description.

Each script can have multiple targets, and each target can have multiple criteria. When an object needs classification, the script processes the targets in turn. Within each target, the script processes criteria sequentially. When it finds that the classification criteria for a target match, it returns the target to the SAE. If the script does not find any targets that can be matched, the classifier engine returns a no match message to the SAE.

Because classification scripts examine criteria sequentially as the criteria appear in the script, you should put more specific criteria at the beginning of the script and less specific criteria at the end of the script.

### Interface Classification Scripts

When a subscriber's IP interface comes up on the router, the router sends the subscriber's login and interface information to the SAE. For example, the router might send the following information:

```
IP address=0.0.0.0
Virtual router name=default@erx5_ssp58
Interface name=FastEthernet3/1.1
PPP login name (PPP)=pebbles@virneo.net
User IP address (PPP)=192.168.55.5
Interface speed=100000000
Interface description=P3/1.1
Interface alias=1st pppoe int
RADIUS class=null
```

The SAE invokes the interface classification script and provides to the script the information that it received from the router. The script engine matches the information sent from the router to the criteria in the interface classification script. The script examines each criterion in sequential order to find a match.

- If it finds a match, the script processing stops, and the target for that criterion is returned to the SAE. The target is the distinguished name (DN) of a policy group in the directory. This policy group is the default policy. The SAE installs the policy on the interface and begins managing the interface.

- If it does not find a match, the script sends a no match message to the SAE. The SAE does not manage the interface; that is, the policies installed through RADIUS or command-line interface (CLI) remain in effect. The SAE does not install policies, and the JUNOSe router does not send reports for this interface anymore.

### *Subscriber Classification Scripts*

When the SAE begins managing an interface, it determines whether a subscriber is associated with the interface by running the subscriber classification script. The SAE also runs the subscriber classification script when certain login events occur. See *Login Events* on page 16 for a description of login event types.

To find the matching subscriber profile, the SAE uses interface information that it received from the router when the interface became operational (for example, virtual router name, interface name, interface alias). It also uses login information that it received from the router when the subscriber attempted to log in (for example, interface name, subscriber IP address, login name, or login event type).

When the SAE runs the subscriber classification script, the script engine matches the information sent from the router to the criteria in the subscriber classification script. The script examines each criterion in sequential order to find a match.

- If it finds a match, the script processing stops, and the target for the matching criterion is returned to the SAE. The target is an LDAP query that uniquely identifies a subscriber entry in the directory. The SAE loads the subscriber entry from the directory and uses the entry to create a subscriber session in memory.

- If it does not find a match, the script sends a no match message to the SAE. The SAE does not load a subscriber session onto the interface, and services cannot be activated on the interface.

### *DHCP Classification Scripts*

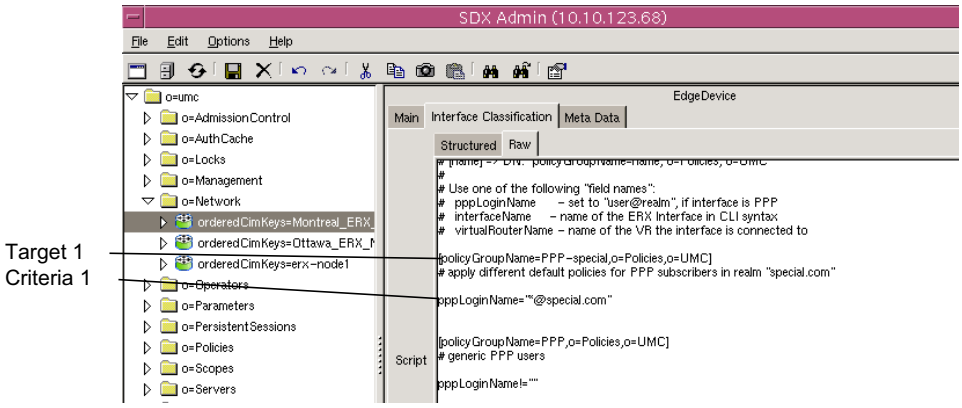
DHCP classification scripts choose DHCP profiles. See *Assigning DHCP Addresses to Subscribers* on page 132 for information about how DHCP classification scripts are used.

## Configuring Classification Scripts

Classification scripts are organized into sections. Each section has a target and one or more classification criteria. The general layout of a classification script is that targets are enclosed in square brackets ([ ]) and precede their criteria:

```
[target-1] (see Figure 20)
  criteria-1 (see Figure 20)
  criteria-2

[target-2]
  criteria-1
  criteria-2
```

**Figure 20: Target and Criteria Structure**




### *Classification Targets*

A target is the result of the classification script that gets returned to the SAE. There are two types of targets:

- - (single dash)—Interpreted as no match. If the criteria of this target are matched, a no match message is returned to SAE. You can use this target to exclude certain patterns or to shortcut known nonmatches. To speed up processing, you could use this option to specify interfaces that you do not want the SAE to manage.

- * (asterisk)—Interpreted as the start of a script target. The complete content of the script target is interpreted when the classifier is initially loaded. The script target can contain definitions of custom functions, which can be called during the matching process. Because you can insert arbitrary code into a script target, you can use the classification script to perform arbitrary tasks.

#### Target Expressions

A target can contain expressions. These expressions can refer to an object in the SAE's memory or configuration, to specific matching criteria, or to another function or script.

Suppose the classification object in a subscriber classifier contains a field called userName. The classifier target uniqueId = <- userName -> is expanded to contain the actual content of the userName field before it is returned to the SAE; for example, for userName = juser, uniqueId = juser is returned.

Target expressions are enclosed in angle brackets and hyphens; for example, <-retailerDn->. The classifier expands expressions before it returns the target to the SAE. The expression is interpreted by an embedded Python interpreter and can contain variables and Python operations. In the simplest case an expression can be a single variable that is replaced with its current contents. Available variable names are all fields of the object passed to the classifier and names created with regular expression matching.

Because a scripting interpreter interprets expressions, more complex operations are possible. Examples are:

- Indexing—var[index] return the element index of a sequence. The first element is at index 0.
- Slicing—var[start : end] create a substring of the variable *var* starting at index startup to, but not including, index end; for example, var = Hello, var[2:4] = ll

## *Classification Criteria*

You organize classification criteria by putting one criterion per line, and joining a criterion with the previous criterion by:

- OR if the line does not contain a prefix or if it is prefixed with a | (pipe) character. A criterion joined by OR is examined only if the previous conditions have not produced a positive match. If any of the criteria joined by OR matches, the target is selected.
- AND if the line is prefixed with an & (ampersand) character. A criterion joined by AND is examined only if the previous condition matches.

You can use glob or regular expression matching to configure each target's criteria.

### Glob Matching

Glob matches are of the form:

```
field = match
or
field != match
```

where match is a pattern similar to UNIX filename matching. Glob matches are case insensitive. "field != match" is true, if field = match is not true.

- *—Matches any substring
- ?—Matches any single character
- [range]—Matches a single character in the specified range. Ranges can have the form a-z or abcd.

- [!range]—Matches a single character outside the specified range
- C—Matches the single character c

The available field names are described for the specific classifiers. Examples are:

- interfaceName = fastEthernet3/0 # match the string “fastEthernet3/0” directly
- interfaceName = fast*3/1 # match any string that starts with “fast” and ends with “3/1”
- interfaceName = fast*3/1.* # start with “fast”, contains “3/1.” arbitrary ending
- interfaceName = fast*3/[2-57] # start with “fast”, contains “3/” followed by 2,3,4,5 or 7

### Regular Expression Matching

Regular expression matches are of the form:

```
field =~ re
or
field !~ re
```

where *field !~ re* is true if *field =~ re* is not true. The regular expression is *re*. For a complete description of the syntax, see:
http://www.python.org/doc/2.0/lib/re-syntax.html

You can group regular expressions with pairs of parentheses. If such an expression matches, the contents of the groups are made available for target expressions. Group number *n* is available as G[*n*], where *n* is the number of the opening parenthesis of the group. You can also name groups by using the special notation (?P < name > ...).

Examples:

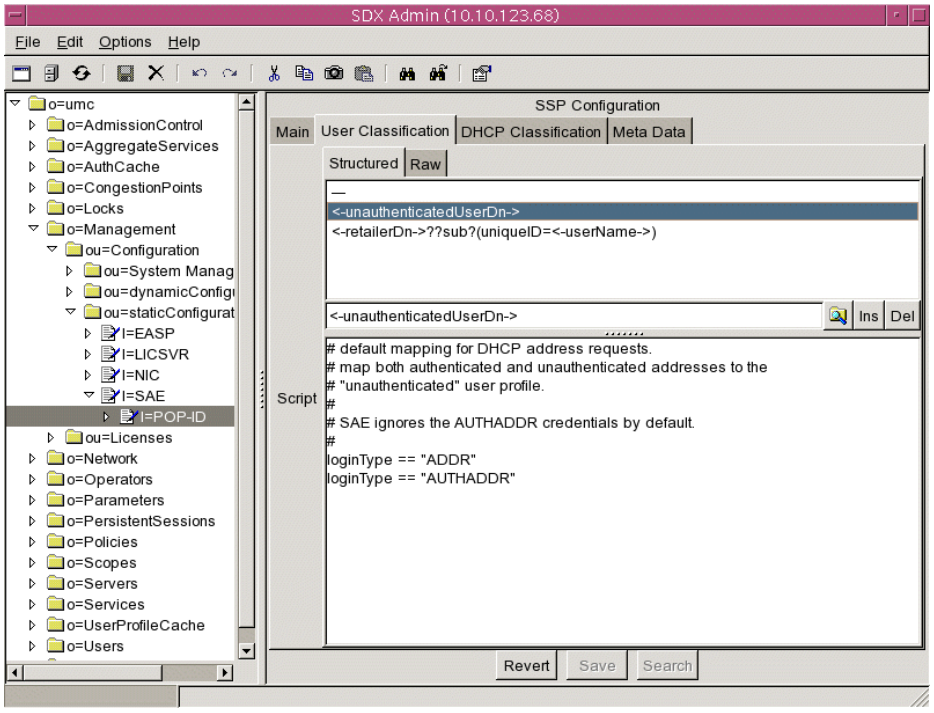
```
ifAlias =~ "SSP(.*)"
# match a string starting with "SSP". The remainder is stored
# in the variable "G[1]"

ifAlias =~ (?P<dn>name=(?P<name>[^,]*).*)
        # match a string starting with “name=“. The whole match is
        # stored in the variable “dn”. A submatch which does not
        # contain any “,”-characters and starts after “name=“
        # is stored in variable “name”
```

## *Configuring Targets in Structured View*

You can create and modify classification scripts with SDX Admin. SDX Admin provides two views of classification scripts—structured and raw. You can switch between the two views at any time and make changes in either view.

Figure 21 shows the structured view of a subscriber classification script.

**Figure 21: Classification Script—Structured View**

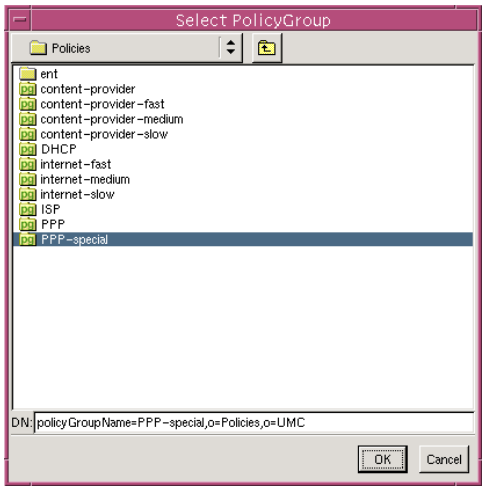



The targets are displayed in the first field. The first entry in the target list (*---*) corresponds to the (unnamed) header section of the classification script. It always exists as the first entry; you cannot delete the target or insert a target in front of it.

To reorder targets, drag a target inside the target list. To edit a target, select the target, which copies the target into an edit field and shows the classification criteria in the Script field. You can then edit the target, or you can use the three buttons to the right of the target editing field to do the following:

-  —Opens a dialog box that you can use to select the distinguished name (DN) of an object in the directory. Figure 22 shows the dialog box for interface classification scripts; it contains the DNs of existing policy groups. Subscriber classification scripts display the DNs of objects in the *o = Users* directory. Dynamic Host Configuration Protocol (DHCP) classification scripts display the DNs of cached DHCP profiles.

**Figure 22: Select PolicyGroup Dialog Box**




- Ins—Inserts a new target after the highlighted target (or at the end if no target is selected)
- Del—Deletes the highlighted target

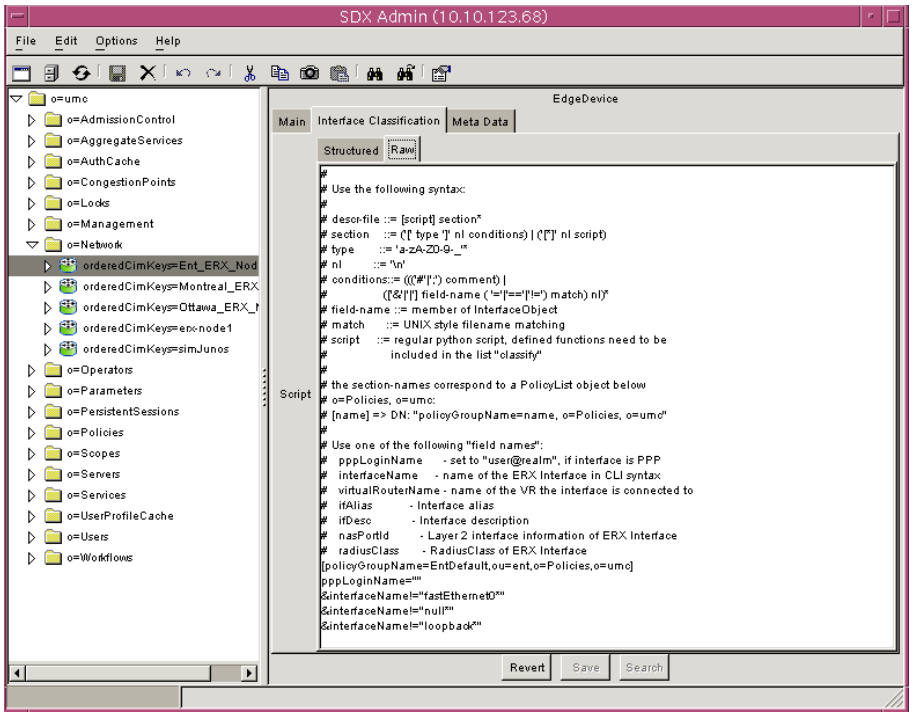
### *Configuring Criteria in Structured View*

Select the target for which you want to configure criteria. SDX Admin displays the classification criteria for the target in the Script field. You can edit the criteria directly in the Script field.

### *Configuring Targets and Criteria in Raw View*

Figure 23 shows the raw view of a classification script. When you are in the raw view, you can copy and paste the contents of a classification script to another object in the directory.

**Figure 23: Classification Script—Raw View**




## Testing Subscriber and Interface Classification Scripts

SAE Web Admin provides a classifier tester that you can use to test subscriber and interface classification scripts. It contains a form that holds the classification script and a form for defining the fields of a subscriber classification context or an interface object.

The test first compiles the classification script. Then it creates a classification context object based on user input, and it invokes the classification script on this context. For subscriber classification scripts, the returned LDAP query is executed.

The output of the test page contains:

- Any compilation or classification errors
- The return value of the classification script; that is, the DN of a policy list or an LDAP query for loading a subscriber profile
- The object returned by the LDAP query (or an error message if the query did not return a unique object)

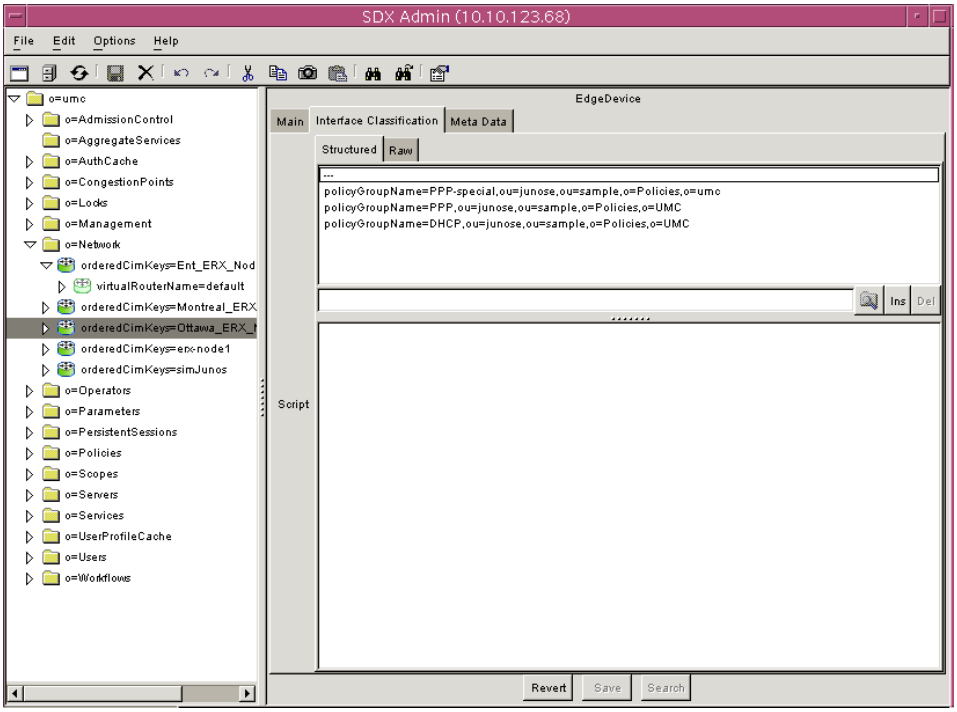
## Classifying Interfaces

To define interface classification scripts with SDX Admin:

1. In the SDX Admin navigation pane, access a router object in *o = network, o = umc*.

2. Click the **Interface Classification** tab.

   The following pane appears.




3. Use the information in *Selecting Interface Classification Criteria* on page 104 and *Configuring Interface Classification Targets* on page 106 to configure an interface classification script.

### *Selecting Interface Classification Criteria*

Interface classification criteria define match criteria that are used to find a policy group. Use the fields in this section to define classification criteria.

#### *broadcastAddr*

- Interface broadcast address.
- Value—Valid broadcast address format
- Example—broadcastAddr.hostAddress = “255.255.255.255”

***ifAlias***

- Description of an interface.
- Value—Interface description that is configured on the router. For JUNOSe routers, it is the description configured with the **interface description** command.
- Example—ifAlias = "1st pppoe int"

***ifDesc***

- Alternate name of the interface that is used by SNMP. This name is a system-generated name.
- Value
  - On a JUNOSe router, the format of the description is

    ip<slot>/<port>.<subinterface>

  - On the JUNOS routing platform, ifDesc is the same as interfaceName.
- Example—ifDesc = "IP3/1.1"

***interfaceName***

- Name of the interface.
- Value
  - Name of the interface in your router CLI syntax
  - FORWARDING_INTERFACE for routing instance (used by traffic mirroring)
- Example—For JUNOSe routers: interfaceName = "fastethernet6/0.1"

  For JUNOS routing platforms: interfaceName = "fe-0/1/0.0"

  For forwarding interface: interfaceName = "FORWARDING_INTERFACE"

***ipAddress***

- Interface IP address.
- Value—Valid IPv4 IP address format
- Example—ipAddress = "10.10.30.1"

***ipMask***

- Interface network mask.
- Value—Valid IPv4 IP network mask format
- Example—ipMask = "255.255.255.255"

***mtu***

- Maximum transfer unit configured on the interface.
- Value—32-integer value
- Example—mtu = "1492"

***nasPortId***

- Port identifier of an interface.
- Value—Includes interface name and additional layer 2 information
- Example—nasPortId = “fastEthernet 3/1” (There is a space between fastEthernet and slot number 3/1 in the nasPortId.)

***pppLoginName***

- Login name for PPP subscribers.
- Value—Login name in the format username@domain
- Example—pppLoginName = “pebbles@virneo.net”

***radiusClass***

- RADIUS class attribute.
- Value—RADIUS class name
- Example—radiusClass = “Premium”

***serviceBundle***

- Content of the vendor-specific RADIUS attribute for the service bundle.
- Value—Name of a service bundle

***userIpAddress***

- Subscriber IP address (PPP only).
- Value—valid IPv4 address
- Example—userIpAddress = “192.168.30.15”

***virtualRouterName***

- Name of the virtual router or routing instance.
- Value—For JUNOSe routers: name of the virtual router in the format vrname@hostname

  For JUNOS routing platforms: name of the routing instance
- Example—virtualRouterName = “default@erx5”

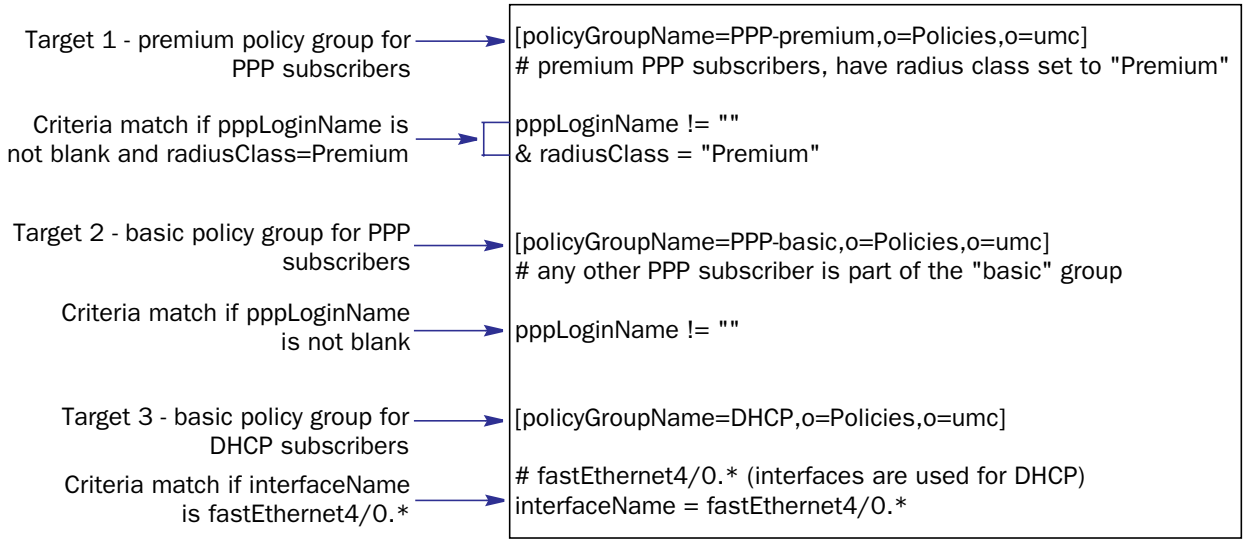
## *Configuring Interface Classification Targets*

The targets of the interface classification scripts are DNs of policy group objects defined in the directory. For example, policyGroupName = DHCP, o = Policies, o = umc

### *Example: Managing Interfaces for Premium and Basic PPP and DHCP Subscribers*

In this scenario, the router manages two types of PPP interfaces—DHCP subscriber interfaces and static IP interfaces. The fastEthernet4/0.1 to fastEthernet4/0.999 interfaces are VLAN interfaces used to terminate DHCP subscribers.

The service provider has separated the PPP subscribers into a premium subscriber group and a basic subscriber group. These groups are distinguished by a different set of default policies applied to the PPP interface. The RADIUS class attribute in the RADIUS profile for premium subscribers is set to Premium. The interface classification script for this scenario is:




The script is processed as follows:

1. If pppLoginName is not blank and radiusClass is Premium, the PPP-premium policy group is sent to the SAE, and script processing stops.

2. If script processing proceeds and pppLoginName is not blank, the PPP-basic policy group is sent to the SAE, and script processing stops.

3. If script processing proceeds and interfaceName is fastEthernet 4/0.0 through fastEthernet 4/0.999, the DHCP policy group is sent to the SAE, and script processing stops.

### *Example: Managing Specific Interfaces*

This example causes the SAE to load the DHCP policy group on IP interfaces on Fast Ethernet modules in slot 3/port 1, slot 1/port 1, or any port on slot 2. The SAE then manages these interfaces.

```
[policyGroupName=DHCP,o=Policies,o=umc]
interfaceName=FastEthernet3/1
interfaceName=FastEthernet1/1
interfaceName=FastEthernet2/*
```

### *Example: Managing Interfaces by Using the Interface Description*

This example causes the SAE to load the DHCP policy group on any interface where the ifAlias starts with DHCP-subscribers.

```
[policyGroupName=DHCP,o=Policies,o=umc]
ifAlias="DHCP-subscribers*"
```

For this approach, you will need to use the **ip description** command to configure interface aliases that begin with DHCP-subscribers for all interfaces that support DHCP subscribers.

## Classifying Subscribers

Changes that you make to subscriber classification scripts do not affect subscriber sessions that are already established. One effect of this behavior is that static IP subscriber sessions are not closed if the classification script is changed in a way that would no longer cause the SAE to load a profile for certain subscribers.

On JUNOSe routers that use the COPS-PR or COPS XDR router drivers, you can create a subscriber session for the router interface to start services such as script services and aggregate services. The SAE creates the router interface, but does not install any policies on it. You can create a subscriber classification rule, but not an interface classification rule for this interface.

To define subscriber classification scripts with SDX Admin:

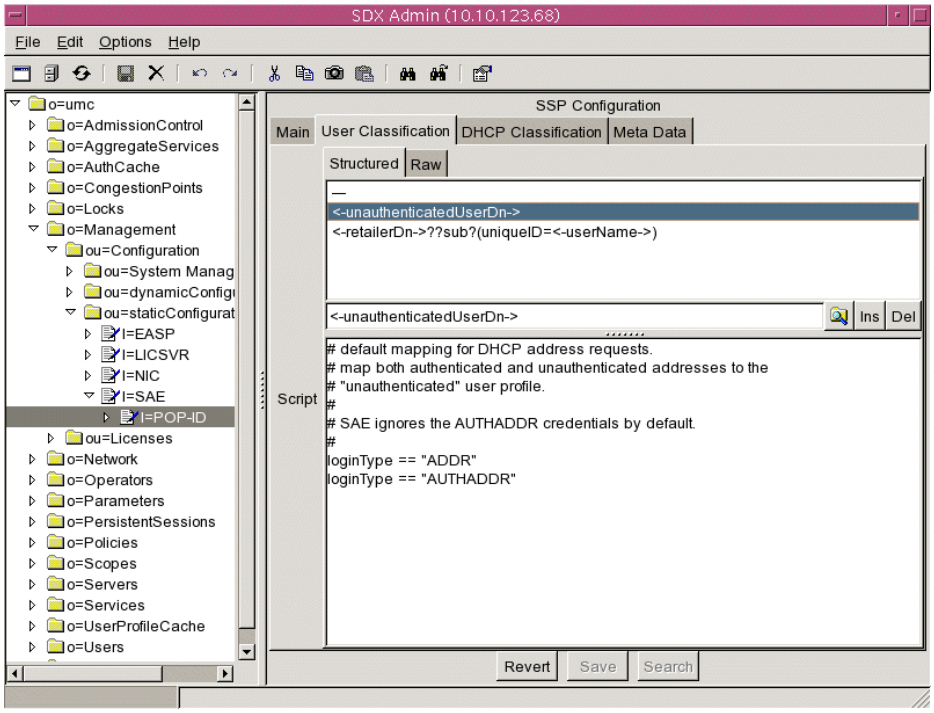
1. In the SDX Admin navigation pane, access the SAE object *I = SAE, ou = staticConfiguration, ou = configuration, o = management, o = umc*.

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

   The SSP Configuration pane appears.

3. Click the **User Classification** tab.

The following screen appears.




Use the information in *Selecting Subscriber Classification Criteria* on page 109 and *Configuring Subscriber Classification Targets* on page 114 to configure the subscriber classification script for an SAE object.

## Selecting Subscriber Classification Criteria

Subscriber classification criteria define match criteria that are used to find the subscriber profile. Use the fields in this section to define classification criteria.

### dhcp

- DHCP options. See *Sending DHCP Options to the JUNOSe Router* on page 112.

### domainName

- Domain name of the subscriber.
- Value—Valid domain name
- Example—domainName = “isp99.com”

***ifAlias***

- Description of the interface.
- Value—Interface description that is configured on the router. For JUNOSe routers, it is the description configured with the **interface description** command
- Example—ifAlias = “dhcp-subscriber12”

***ifDesc***

- Alternate name for the interface that is used by SNMP. This name is a system-generated name.
- Value
  - On a JUNOSe router, the format of the description is

    ip<slot>/<port>.<subinterface>

  - On the JUNOS routing platform, ifDesc is the same as interfaceName.
- Example—ifDesc = “IP3/1.1”

***interfaceName***

- Name of the interface.
- Value
  - Name of the interface in your router CLI syntax
  - FORWARDING_INTERFACE for routing instance (used by traffic mirroring)
- Example—For JUNOSe routers: interfaceName = “fastEthernet6/0”

  For JUNOS routing platforms: interfaceName = “fe-0/1/0.0”

  For forwarding interface: interfaceName = “FORWARDING_INTERFACE”

***loginName***

- Name to be used to create a loginName attribute for a subscriber session for JUNOSe interfaces that are not otherwise assigned a loginName when a session starts, such as unauthenticated DHCP addresses, unauthenticated IP interfaces (that are not using PPP connections), or core-facing interfaces.

  The loginName can also be used to identify a subscriber session through the SAE CORBA remote API.

- Value—Name in the form subscriber@domain
- < Login name >
- Guideline—The format is not defined. A loginName can be of form subscriber, domain\subscriber, subscriber@domain, or as otherwise defined by the login setup of the operator.
- Example—idp@idp

***loginType***

- Type of subscriber session to be created.
- Value—One of the following login types:
  - ASSIGNEDIP—For assigned IP subscribers. Triggered when an application accesses a subscriber object for an assigned IP subscriber that is not currently loaded into memory. (Supported on JUNOSe routers.)
  - AUTHINTF—For authenticated interface login requests. Triggered when a login Name is reported together with the interface, such as authenticated PPP or autoconfigured ATM interface, by means of the **subscriber** command. (Supported on JUNOSe routers.)
  - INTF—For unauthenticated interface login requests. Triggered when an interface comes up and the interface classification script determines that the SAE should manage the interface. (Supported on JUNOS routing platforms and JUNOSe routers.)
  - ADDR—For unauthenticated address login requests. Triggered when the DHCP server in the JUNOSe router provides an unauthenticated IP address. (Supported on JUNOSe routers.)
  - AUTHADDR—For authenticated address login requests. Triggered when the DHCP server in the JUNOSe router provides an authenticated IP address. (Supported on JUNOSe routers.)
  - PORTAL—Triggered when the portal API is invoked to log in a subscriber. (Supported on JUNOS routing platforms and JUNOSe routers.)
- Example—loginType = “AUTHADDR”

***macAddress***

- String representation of the DHCP subscriber media access control (MAC) address.
- Value—Valid MAC address
- Example—macAddress = “00:11:22:33:44:55”

***nasPortId***

- Port identifier of an interface.
- Value—Includes interface name and additional layer 2 information
- Example—nasPortId = “fastEthernet 3/1” (There is a space between fastEthernet and slot number 3/1 in the nasPortId.)

***radiusClass***

- RADIUS class used for authorization.
- Value—RADIUS class name
- Example—radiusClass = “Premium”

***retailerDn***

- DN of the retailer object. The object is found when the domain name is mapped to a retailer object in LDAP.
- Value—DN of a retailer

***serviceBundle***

- Content of the vendor-specific RADIUS attribute for the service bundle.
- Value—Name of a service bundle
- Example—serviceBundle = “goldSubscriber”

***unauthenticatedUserDn***

- DN of the unauthenticated subscriber profile (usable for target expressions only).
- Value—DN of a subscriber profile

***userName***

- Name of the subscriber.
- Value—Subscriber name without the domain name
- Example—userName = “peter”

***virtualRouterName***

- Name of the virtual router or routing instance.
- Value—For JUNOSe routers: name of the virtual router in the format vrname@hostname

  For JUNOS routing platforms: name of the routing instance
- Example—virtualRouterName = “default@e_series5”

### Sending DHCP Options to the JUNOSe Router

Subscriber classification scripts support DHCP options conveyed through COPS. When COPS reports an address, the JUNOSe router sends DHCP options received for DHCP requests for that address. The DHCP options are available in the subscriber classification context for selecting the subscriber profile to load.

The fields in Table 14 are in the user classification context of subscriber classification scripts.

**Table 14: DHCP Options in UserClassificationContext Field**

| DHCP Option | UserClassificationContext Field | Comments |
|---|---|---|
| giAddr | dhcp.giAddr | Relay agent gateway address |
| Option 82 data | dhcp.getOption(82) | Content is accessible with getSubOptions() |
| Client ID | dhcp.getOption(61).getString() | |
| Lease time | dhcp.getOption(51).getInt() | |
| Client requested parameter list | dhcp.getOption(55).getBytes() | |
| Domain name sent to client | dhcp.getOption(12).getString()<br>dhcp.getOption(15).getString() | 12 = HostName<br>15 = DomainName |
| DNS server address(es) sent to client | dhcp.getOption(6).getIpAddresses() | |
| Subnet mask | dhcp.getOption(1).getIpAddress() | |
| NetBios name server address(es) sent to client | dhcp.getOption(44).getIpAddresses() | |
| NetBios node type | dhcp.getOption(46).getBytes() | |
| Default router address(es) sent to client | dhcp.getOption(3).getIpAddresses() | |

The DHCP options are accessible for the subscriber classification script with the following syntax:

```
dhcp.giAddr = “match”

# interpret option 61 as string
dhcp[61].string = “match”

# interpret option 1 (subnet) as dotted decimal IP
dhcp[1].ipAddress = “match”

# option 82, suboption 1, interpreted as string
dhcp[82].subOptions[1].string = “match”
```

The received DHCP options are also stored in the UserSession and are available through the portal API (method User.getDhcpOptions).

### *Configuring Subscriber Classification Targets*

The target of the subscriber classification script is an LDAP search string. The search string uses a syntax similar to an LDAP URL (see RFC 2255—The LDAP URL Format (December 1997)). The syntax is:

```
baseDN [ ? [ attributes ] [ ? [ scope ] [ ? [ filter ] ] ] ]
```

- baseDN—Distinguished name of object where the LDAP search starts
- attributes—Can be used to override attributes in the loaded LDAP object. For example, for static IP subscribers the SAE must learn the IP address assigned to a particular subscriber. This address is defined in the ipAddress attribute of the subscriber profile. A target of the form baseDN?ipAddress=<-function(interfaceName)-> invokes function after the subscriber profile is loaded from LDAP and sets the IP address to the return value of function. The function is defined in the subscriber classification script, and can be used for a variety of things; for example, to query an external database.
- scope—Scope of search in the directory
  - base—Is the default, searches the base DN only.
  - one—Searches the direct children of the base DN.
  - sub—Searches the complete subtree below the base DN.
- filter—Is an RFC 2254–style LDAP search filter expression; for example, (uniqueId=<-userName->). See RFC 2254—The String Representation of LDAP Search Filters (December 1997).

With the exception of baseDN all the fields are optional.

The result of the LDAP search must be exactly one directory object. If no object or more than one object is found, the subscriber session is terminated.

### *Example: Subscriber Classification Scripts for Static IP Subscriber*

In cases such as bridged 1483 DSL with a single subscriber, you can write the subscriber classification script so that it loads a specific subscriber profile. If the interface is matched to a subscriber profile, a subscriber session is immediately established. An SAE application (for example, a portal) can still force the subscriber with this subscriber profile to perform a Web login.

One way to achieve the mapping of subscriber interface to subscriber profile is to provision the assigned interface name in the associated subscriber profile in LDAP. In this case the subscriber classification script can include a rule like this:

```
[retailerName=default,o=Users,o=umc??sub?(interfaceName=<-interfaceName->)]
# all fastEthernet interfaces are connected to static IP subscriber
loginType = INTF
& interfaceName = fastEthernet*
```

Another way may include a special encoding of the interface alias (ifAlias) field of the subscriber interface. This encoding must then be provisioned when the interface for the subscriber is provisioned. In this example, the encoding SSP-username is chosen for ifAlias; for example, for subscriber juser the interface alias would be set to SSP-juser. The match is performed with a regular expression, which separates the user ID from the ifAlias prefix.

```
[retailerName=default,o=Users,o=umc??sub?(uniqueID=<-userId->)]
loginType = INTF
& ifAlias =~ SSP-(?P<userId>.*)
```

### *Example: Subscriber Classification Scripts Using a Subscriber Group*

To support scenarios where SAE has no access to the subscriber database, SAE can load anonymous profiles for groups of subscribers. The following example loads a particular subscriber profile when subscribers of domain another-isp.com log in.

```
[uniqueID=anon,ou=default,retailerName=another-isp,o=Users,o=umc]
domainName = another-isp.com
```

### *Example: Subscriber Classification Scripts for Enterprise Subscribers*

For enterprise subscribers, you can create one general subscriber classifier script that matches a unique subscriber profile to each managed router interface. The subscriber profile is the access subscription that represents an Internet access in an enterprise. The following examples show two approaches to creating the general classifier script. You can use one of these strategies or a combination of strategies.

#### Matching on the Interface Name

In this scenario, you configure the interface name field in the access subscription for the site to match an interface on the router. The format for the interface name could be: interfaceName@virtualRouterName@routerName. You then create a classification script that searches for subscriber profiles that match a specific interface. For example:

```
[ou=Managed CPE,retailerName=Retailer-Two,o=Users,o=UMC??sub?\
(interfaceName=<-interfaceName->@<-virtualRouterName->)]
loginType = INTF
& interfaceName = "fe*"
```

#### Matching on the Interface Alias

For JUNOSe routers, you can configure the interface description on the router in a format that the classifier script can match to the interface alias in an access subscription. In a simple case, you can configure the interface description only for interfaces that terminate a managed CPE, and match them to the interface alias in the directory. The subscriber classifier could be configured as follows:

```
[ou=Managed CPE,retailerName=Retailer-Two,o=Users,o=UMC??sub?\
(interfaceAlias=<-ifAlias->)]
ifAlias != ""
```

### *Example: Subscriber Classification Scripts For a Wholesaler/Retailer Scenario*

In a wholesaler/retailer scenario, where the wholesaler owns the SAE but the retailer authenticates subscribers using RADIUS, it is possible to use a RADIUS vendor-specific attribute (VSA) (serviceBundle = Juniper(4874) #31) to send information from the RADIUS profile to the SAE. The subscriber classification script is then used to load a different subscriber profile with different subscriptions based on information stored in the RADIUS database of the retailers; for example:

```
[uniqueID=<-serviceBundle->,ou=default,retailerName=another-isp,o=Users,o=umc]
domainName = another-isp.com
& serviceBundle != ""
```

Alternatively, the target can be written as an LDAP search, taking advantage of the domain name–to–retailer object mapping of the SAE; for example,

```
[<-retailerDn->??sub?(uniqueId=<-serviceBundle->]
serviceBundle != ""
```

### *Example: Creating Router Interface Subscriber Session*

Aggregate services or script services can be activated on a router instead of an interface or DHCP address. On JUNOSe routers that use the COPS-PR or COPS XDR router driver, the SAE automatically creates a router interface; and then a subscriber session as specified by the subscriber classification script.

For example, the following script searches for a router profile in the directory under ou = routers, retailerName = default, o = Users, o = umc, with a routerName that matches the virtual router name (such as default@erx-node1).

```
[ou=routers,retailerName=default,o=Users,o=UMC??one?(routerName=<-virtualRouter
Name->)]
interfaceName = Router
```

### *Example: Activating Services for a Group of Subscriber Sessions*

A subscriber classification script can assign a shared subscriber profile and a login name to a subscriber session for a group of interface subscriber sessions. The following example assigns the login name idp@idp to subscriber sessions for JUNOSe interfaces that have core specified as the ifAlias (as configured on the JUNOSe router).

```
[routerName=idp,ou=interfaces,retailername=SP-IDP,o=Users,o=UMC?loginName=idp
@idp]
# core facing interfaces on JUNOSe routers in JUNOSe POPs
ifAlias=="core"
```

You can use this type of subscriber classification to activate a service for a group of interface subscriber sessions that are to be treated the same. For example in the configuration for an aggregate service, a fragment service could be created for all subscriber interface sessions on interfaces identified by the ifAlias core on a virtual router. The subscriber reference expression in the configuration for the fragment service would reference the virtual router name and the login name, such as vr = “<- virtualRouterName ->”, login_name = “idp@idp.”

You can also use the SAE CORBA remote API to get lists of the subscriber sessions that share the same login name.

## Classifying DHCP Subscribers

DHCP classification scripts are stored in the directory in the dhcpProfileClassification attribute of the umcConfiguration object class. They contain fields set by the address request and authorization response.

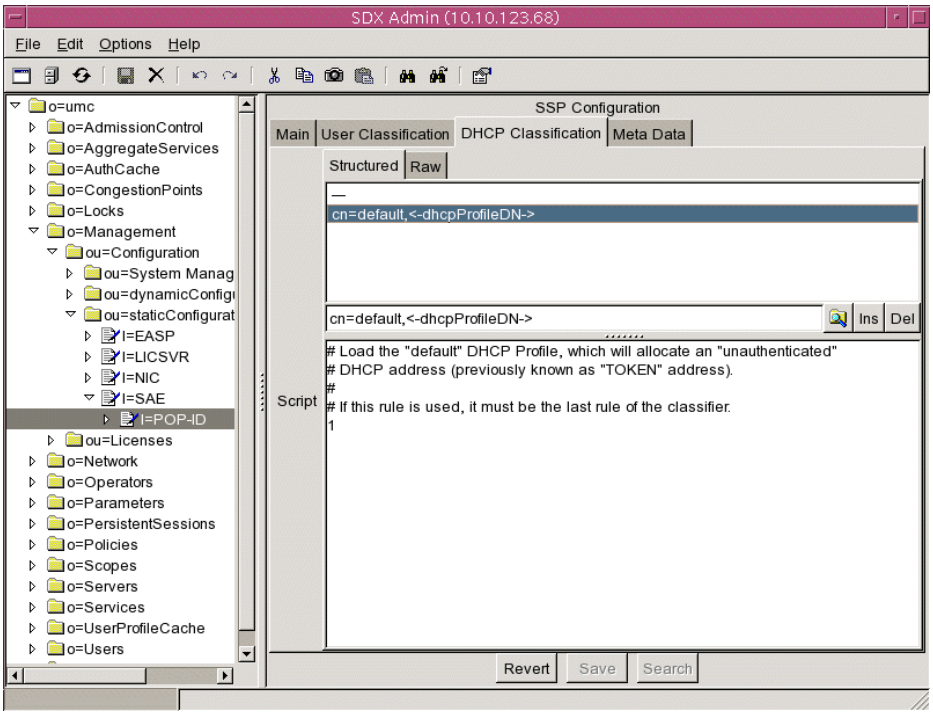
To configure DHCP classification scripts with SDX Admin:

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.

3. Select the **DHCP Classification** tab.

   The structured view of the DHCP classification configuration appears.




Use the information in *Selecting DHCP Classification Criteria* on page 118 and *Configuring DHCP Classification Targets* on page 119 to configure the DHCP classification script for an SAE object.

### Selecting DHCP Classification Criteria

DHCP classification criteria define match criteria that are used to find the DHCP profile. Use the fields in this section to define DHCP classification criteria.

***authVirtualRouterName***

- Name of JUNOSe virtual router that is set by an authorization plug-in through the authorization response.
- Value—Name of the virtual router in the format vrname@hostname

***dhcp***

- DHCP options. See *Setting DHCP Parameters with DHCP Options* on page 120.

***dhcpProfileDN***

- Search base for DHCP profiles. The DN can be used in target expressions.
- Value—DN of DHCP profile

***interfaceName***

- Name of the interface where the DHCP discover message was received.
- Value—Name of the interface in your router CLI syntax
- Example—interfaceName = fastEthernet6/0

***ifAlias***

- Description of the interface where the DHCP discover request was received.
- Value—Interface description that is configured on the router. For JUNOSe routers, it is the description configured with the **interface description** command
- Example—ifAlias = “dhcp-subscriber12”

***ifDesc***

- Alternate name for the interface where the DHCP discover request was received. This is a system-generated name that is used by SNMP.
- Value
  - On a JUNOSe router, the format of the description is:

    ip<slot>/<port>.<subinterface>

  - On the JUNOS routing platform, ifDesc is the same as interfaceName.

***macAddress***

- MAC address of the DHCP client that appears in DHCP request.
- Value—Valid MAC address
- Example—macAddress = “00:11:22:33:44:55”

***nasPortId***

- Port identifier of an interface.
- Value—Includes interface name and additional layer 2 information
- Example—nasPortId = “fastEthernet 3/1” (There is a space between fastEthernet and slot number 3/1 in the nasPortId.)

***poolName***

- IP address pool name that is set by an authorization plug-in through the authorization response.
- Value—Name of an address pool configured on the JUNOSe router

***virtualRouterName***

- Name of the virtual router.
- Value—Name of the virtual router in the format vrname@hostname

## *Configuring DHCP Classification Targets*

The target of the DHCP classification script uses a syntax similar to an LDAP URL. With the exception of baseDN, all fields are optional. The syntax is:

```
baseDN [ ? [ attributes ] [ ? [ scope ] [ ? [ filter ] ] ] ]
```

- baseDN—DN of object where search starts.
- attributes—Comma-separated list of properties, in the format attribute = <-value->, that allow you to set specific attributes for directory objects that the script finds; see *Selecting DHCP Classification Criteria* on page 118.

  You can use the attribute configuration to override attributes in the directory. For example, to override the IP pool name that is stored in the DHCP profile with the pool name that the authorization plug-in sends, use the attribute statement radiusFramedPool = <-poolName->.
- scope—Scope of search in the directory
  - base—Searches the base DN only; default scope
  - one—Searches the direct subordinates of the base DN (one-level search)
  - sub—Searches all objects subordinate to the base DN
- filter—An RFC 2254–style LDAP search filter expression; for example, (uniqueId = <-userName->). See RFC 2254—The String Representation of LDAP Search Filters (December 1997).

## Selecting DHCP Parameters

The SAE sends a set of parameters to the DHCP server in the JUNOSe router. The DHCP server determines the IP address offered, as well as the options sent to the DHCP client. The parameters comprise IP address authorization parameters, as well as parameters stored in a DHCP profile in the directory. Parameters in the DHCP profile override authorization parameters.

For more information about how the SAE handles DHCP subscribers, see:

- Assigning DHCP Addresses to Subscribers on page 132
- DHCP Subscriber Login and Service Activation on page 22

### *Setting DHCP Parameters with DHCP Options*

**NOTE:** JUNOSe routers do not currently support the functionality described in this section. DHCP options and BOOTP options that the SAE sends to the JUNOSe router are ignored.

DHCP servers use DHCP options to configure DHCP clients. The DHCP local server in the JUNOSe router supports a subset of DHCP options. The SAE supports all DHCP options defined in RFC 2132—DHCP Options and BOOTP Vendor Extensions (March 1997) by name. It also supports other options, but you need to specify them by number and type. The DHCP options allow a flexible definition of parameters offered to DHCP subscribers. For example, they allow integration with cable modems or set-top boxes because you can configure options to control the boot sequence of these devices.

You can configure DHCP options in DHCP profiles and in DHCP classification scripts. Table 15 on page 121 lists the name, number, and type of all supported DHCP options. You can use these fields to configure DHCP options.

The following example shows how to specify an option by number and by type. The two statements identify the same option:

```
dhcp[12]

dhcp['host-name']
```

In SDX software earlier than Release 4.2, you had to include the option type in your option definition. For example:

```
dhcp[12].string = HOST
```

You can now write:

```
dhcp[12] = HOST
```

Note that the earlier method of defining options still works in Release 4.2 and later.

**Table 15: DHCP Options Supported on the SAE**

| Option Name | Option Number | Option Type |
|---|---|---|
| subnet-mask | 1 | ip-address |
| time-offset | 2 | int32 |
| routers | 3 | ip-address |
| time-servers | 4 | ip-address |
| ien116-name-servers | 5 | ip-address |
| domain-name-servers | 6 | ip-address |
| log-servers | 7 | ip-address |
| cookie-servers | 8 | ip-address |
| lpr-servers | 9 | ip-address |
| impress-servers | 10 | ip-address |
| resource-location-servers | 11 | ip-address |
| host-name | 12 | string |
| boot-size | 13 | int16 |
| merit-dump | 14 | string |
| domain-name | 15 | string |
| swap-server | 16 | ip-address |
| root-path | 17 | string |
| extension-path | 18 | string |
| ip-forwarding | 19 | int8 |
| non-local-source-routing | 20 | int8 |
| policy-filter | 21 | ip-address |
| max-dgram-reassembly | 22 | int16 |
| default-ip-ttl | 23 | int8 |
| path-mtu-aging-timeout | 24 | int32 |
| path-mtu-plateau-table | 25 | int16 |
| interface-mtu | 26 | int16 |
| all-subnets-local | 27 | int8 |
| broadcast-address | 28 | ip-address |
| perform-mask-discovery | 29 | int8 |
| mask-supplier | 30 | int8 |
| router-discovery | 31 | int8 |
| router-solicitation-address | 32 | ip-address |
| static-routes | 33 | ip-address |
| trailer-encapsulation | 34 | int8 |
| arp-cache-timeout | 35 | int32 |
| ieee802-3-encapsulation | 36 | int8 |
| default-tcp-ttl | 37 | int8 |
| tcp-keepalive-interval | 38 | int32 |

**Table 15: DHCP Options Supported on the SAE (continued)**

| Option Name | Option Number | Option Type |
|---|---|---|
| tcp-keepalive-garbage | 39 | int8 |
| nis-domain | 40 | string |
| nis-servers | 41 | ip-address |
| ntp-servers | 42 | ip-address |
| netbios-name-servers | 44 | ip-address |
| netbios-dd-server | 45 | ip-address |
| netbios-node-type | 46 | int8 |
| netbios-scope | 47 | string |
| font-servers | 48 | ip-address |
| x-display-manager | 49 | ip-address |
| requested-ip-address | 50 | ip-address |
| ip-address-lease-time | 51 | int32 |
| option-overload | 52 | int8 |
| dhcp-msg-type | 53 | int8 |
| server-identifier | 54 | ip-address |
| parameter-request-list | 55 | data-string |
| message | 56 | string |
| maximum-dhcp-msg-size | 57 | int16 |
| renewal-time | 58 | int32 |
| rebinding-time | 59 | int32 |
| vendor-class-identifier | 60 | data-string |
| client-identifier | 61 | data-string |
| nisplus-domain | 64 | string |
| nisplus-servers | 65 | ip-address |
| tftp-server-name | 66 | string |
| bootfile-name | 67 | string |
| mobile-ip-home-agent | 68 | ip-address |
| smtp-server | 69 | ip-address |
| pop-server | 70 | ip-address |
| nntp-server | 71 | ip-address |
| www-server | 72 | ip-address |
| finger-server | 73 | ip-address |
| irc-server | 74 | ip-address |
| streettalk-server | 75 | ip-address |
| streetalk-directory-assistance-server | 76 | ip-address |

## Creating DHCP Profiles

When the SAE receives a DHCP discover request from the router, it uses the client's MAC address to find a DHCP profile in cache or in the directory. If it finds a DHCP profile, the SAE uses the information in the profile to create a discover decision that it returns to the router. The discover decision includes information to select an IP address and DHCP options to configure the DHCP client.

When a DHCP subscriber logs in to the SAE through a Web portal, the SAE registers the subscriber's equipment and creates a cached DHCP profile in the *o = AuthCache* directory. These profiles are keyed by the MAC address of the DHCP client device. They are created by the grantPublicIp or the registerEquipment methods.

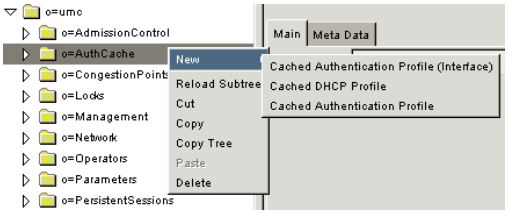
You can also create DHCP profiles manually with SDX Admin or by adding DHCP profile entries to the directory. DHCP profiles are stored in the *o = AuthCache* directory in the dhcpProfile object class. The dhcpProfile object class is subordinate to the cachedAuthenticationProfles object class. Manually created profiles are keyed by the cn (common name) attribute.

For more information about how the SAE handles DHCP subscribers, see:

- Assigning DHCP Addresses to Subscribers on page 132
- DHCP Subscriber Login and Service Activation on page 22

To create a DHCP profile with SDX Admin:

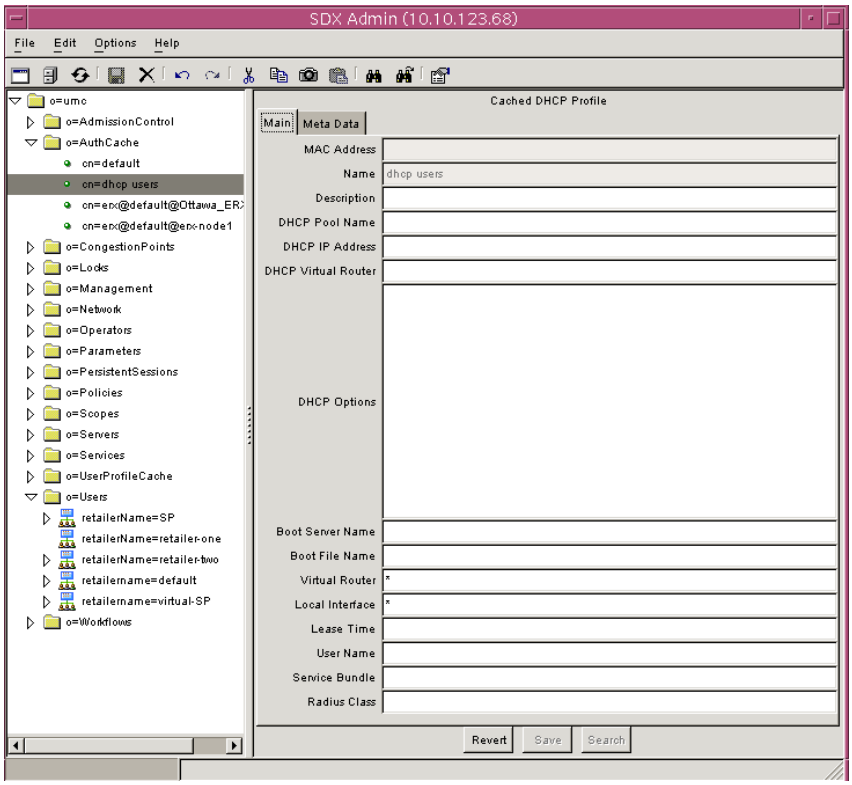
1. Highlight **AuthCache**, and right-click.
2. Select **New > Cached DHCP Profile**.




   The New Cached DHCP Profile dialog box appears.

3. Assign a name to the profile.
4. Click **OK**.

The Cached DHCP Profile pane appears.




5. Fill in the fields as described in this section.

### *MAC Address*

- Naming attribute for system-created DHCP profiles. When a DHCP subscriber first logs in to the SAE, the subscriber's equipment is registered, and the SAE caches the MAC address in the *o = AuthCache* directory. System-created profiles are keyed by MAC address.
- Value—SAE fills in the MAC address
- Default—No value
- Attribute name—macAddress

### *Name*

- Naming attribute for manually created DHCP profiles. Manually created profiles are keyed by MAC address.
- Value—String
- Default—No value
- Attribute name—cn

***Description***

- Text description of the profile.
- Value—String
- Default—No value
- Attribute name—Description

***DHCP Pool Name***

- Name of the IP address pool on the JUNOSe router from which a DHCP address is selected.
- Value—String, optional
- Default—No value
- Attribute name—radiusFramedPool

***DHCP IP Address***

- Fixed IP address that is offered to the DHCP client if the client is part of a network in the configured DHCP pool.
- Value—String, optional
- Default—No value
- Attribute name—radiusFramedIPAddress

***DHCP Virtual Router***

- Name of the JUNOSe virtual router that holds the IP address pool.
- Value—String, optional
- Default—No value
- Attribute name—virtualRouterName

***DHCP Options***

- Defines DHCP options that are used to configure DHCP clients. See *Setting DHCP Parameters with DHCP Options* on page 120 for more information.
- Value—You define DHCP options in the format:

  ```
  option = value [, value...]
  ```

  where option is the option name or number (see Table 15 on page 121) and values are entered based on the type of option:

  - int32, int16, int8—Decimal or hex prefixed by "0x"
  - string—Optionally surrounded by double quotes
  - ip-address—Dotted decimal
  - data-string—Sequence of hex-encoded bytes separated by ":" or a string surrounded by double quotes

To include nonstandard options in a DHCP profile, use the name "option-*nnn*", where *nnn* is the option number, and the value is of type "data-string." That is, either a string surrounded in double quotes, or a sequence of hex-encoded bytes, separated by ":".

- Default—No value
- Attribute name—dhcpOptions

***Boot Server Name***

- Name of the server used to boot the DHCP client.
- Value—String, length < 64
- Default—No value
- Attribute name—dhcpServer

***Boot File Name***

- Name of a boot file used to boot the DHCP client.
- Value—String, length < 128
- Default—No value
- Attribute name—bootFileName

***Virtual Router***

- Name of the JUNOSe virtual router that is used to check the validity of system-created DHCP profiles.
- Value—Name of the virtual router in the format vrname@hostname. An * (asterisk) means that the values for the virtual router are ignored when the cached profile is used. Use an * if you do not know the virtual router to which the subscriber will connect.
- Default—* (asterisk)
- Attribute name—checkVrName

***Local Interface***

- Name of the JUNOSe interface that is used to check the validity of system-created DHCP profiles.
- Value—Name of interface in JUNOSe CLI syntax (for example, fastethernet6/0). An * (asterisk) means that the values for the local interface are ignored when the cached profile is used. Use an * if you do not know the interface to which the subscriber will connect, or you want to allow the subscriber to connect through multiple interfaces. You can also enter expressions of the form @expr = value.
- Default—* (asterisk)
- Attribute name—localInterface

***Lease Time***

- Length of time the supplied IP address is valid.

**NOTE:** This parameter is not currently implemented on the JUNOSe router. The DHCP lease time that the SAE sends to the JUNOSe router is ignored.

- Value—Number of seconds
- Default—No value
- Attribute name—leaseTime

***User Name***

- Name of DHCP user without the domain name.
- Value—String that specifies the information to the left of the @ character in <userName>@<domainName>.
- Default—No value
- Attribute name—userName

***Service Bundle***

- Vendor-specific RADIUS attribute that specifies the SDX service bundle to use.
- Value—String
- Default—No value
- Attribute name—serviceBundle

***Radius Class***

- RADIUS attribute class.
- Value—String that maps to a RADIUS attribute class
- Default—No value
- Attribute name—radiusClass