Chapter 14

Defining and Acquiring Values for Parameters

This chapter provides information about how the SAE acquires values for policies. Topics include:

Parameters and Substitutions

Each subscriber who uses the SRC network must appear in the directory. You do not need to configure a policy for each subscriber, however. You can define a smaller number of policies that contain *parameters*. A parameter is a general definition for a property, such as an IP address, and is analogous to a variable in a computer program.

The SRC software defines some global parameters and system (runtime) parameters in the policy repository. You can also define your own global parameters in the policy repository, your own local parameters in policy groups, and your own local parameters in other specified items, such as services. See *Chapter 8, Overview of Using Local and Global Parameters.*

When the SAE activates a subscription to a service for a subscriber, it constructs an exact policy for that subscriber by obtaining specific values for parameters. The SAE acquires one or more values for each parameter from a number of different sources. These sources can also contain local parameters for which other sources can provide specific values. The SAE selects a value based on a ranking of sources from specific to general. The process of providing a value or a new definition for a parameter is a *substitution*.

One or more sources can define a parameter as fixed. Fixing prevents acquisition of values from more specific sources in the ranking list. For example, if a parameter is fixed in a subscription for a parent subscriber, a subordinate subscriber cannot provide a more specific value for a parameter in the subscription it inherits from the parent. If a parameter is fixed in more than one place, the SAE uses the setting in the source that is classified as more general.

You can fix a parameter without specifying a value. Doing so specifies that the value for the parameter cannot come from a more general source than the one that contains the fixed setting and that a value will be available at some point. For example, you could fix the value of the system parameter interface_speed in the service scope to prevent more specific sources in the ranking list, such as subscribers, from providing a value for this parameter. The SAE could acquire an actual value for this parameter when it starts managing an interface.

The SAE fixes global and system parameters at a set point in the acquisition chain. Consequently, the SAE can acquire values for these types of parameters only from a service scope, from information the SAE obtains when it starts managing an interface, or from the default value in the global parameter definition.

When you are designing policies, services, portals, and applications, you need to consider how you will use substitutions throughout the software. As a simple example, you can define the general settings for a rate limiter in a policy, insert a parameter for a rate in the policy, and provide specific values for the rate in each service that uses this policy. In a more complex example, you can use parameters and substitutions to track the use of a particular service by different departments in an enterprise (see *Example: Parameter Value Substitution* on page 411).

Value Acquisition for Single Subscriptions

When a subscriber has a single subscription to a service, the SAE ranks sources in the following order when it selects values for parameters:

1. The service sessions associated with the subscriber
2. The subscriber's subscription to a service and then the subscriber
3. Each parent subscriber's subscription and then the parent subscriber
4. The value-added services in order of the precedences defined for their associated service scopes
5. The default values for the local parameters in the policy group
6. Fixed settings of all global parameters defined in the policy repository

7. The service scopes, in order of precedence for each of the services. See:

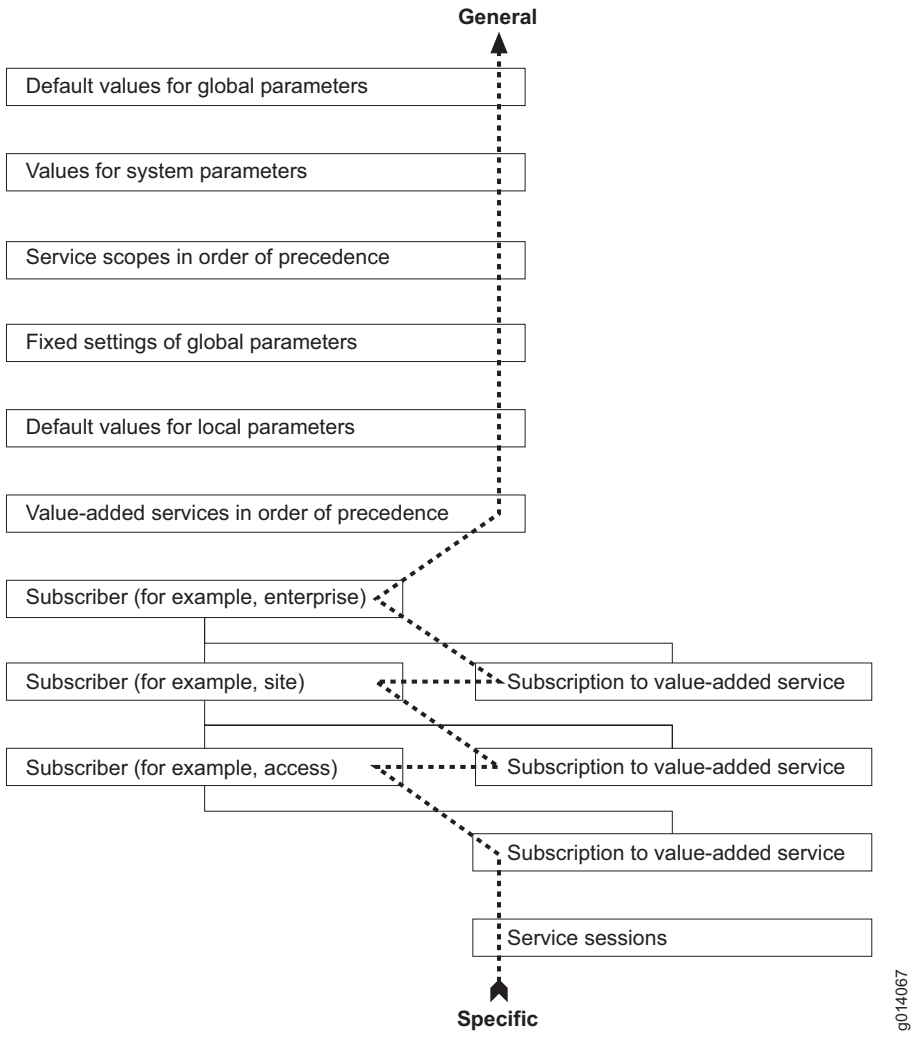
   - *Restricting and Customizing Services for Subscribers* on page 27 in *Chapter 1, Managing Services with the SRC CLI.*
   - *Restricting and Customizing Services for Subscribers* on page 82 in *Chapter 2, Managing Services on a Solaris Platform.*

8. Values for system parameters that are available only when the SAE starts managing the interface (for example, actual bandwidth rates)

9. The default values for global parameters defined in the policy repository

Figure 41 illustrates how the SAE selects values for a subscriber with one subscription to a service.

Figure 41: Value Acquisition for Single Subscriptions



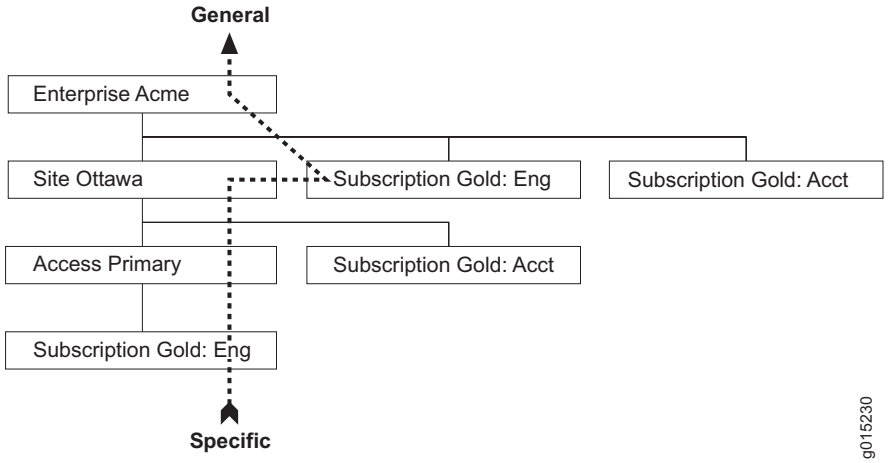
Value Acquisition for Multiple Subscriptions

A subscriber can have multiple subscriptions, each with different service parameters, to the same service. When a subscriber has multiple subscriptions to the same service, each subscription has a different name. The name is determined by the parameters. Different subscribers can have subscriptions with the same name.

As described previously, the SAE considers the subscriptions of parent subscribers when it acquires parameters for the policy of a subordinate subscriber who has one subscription to a service. When acquiring parameters for the policy of a subordinate subscriber who has multiple subscriptions to a service, however, the SAE considers the parent subscriber's subscription only if it has the same name as the subordinate subscriber's subscription.

Figure 42 shows an example that illustrates this concept.

Figure 42: Value Acquisition for Multiple Subscriptions



In this example, an enterprise called Acme contains a site called Ottawa that contains an access called Primary. The access has a subscription called Gold:Eng to the service called Gold; the site has a subscription called Gold:Acct to the same service; and the enterprise has two subscriptions, Gold:Eng and Gold:Acct, to the service.

When the IT manager activates the Gold:Eng subscription for the access, the SAE will consider the parameters in the subscriptions Gold:Eng for the access and the enterprise; however, the SAE will not consider the parameters in the subscriptions called Gold:Acct for the site or the enterprise.

The SAE acquires parameters from other sources in the same way whether the subscriber has multiple subscriptions to a service or a single subscription to a service (see *Value Acquisition for Single Subscriptions* on page 398).

Defining Parameters

You can define parameters for different items in the SRC software. Depending on the item, you can define parameters with the SRC CLI, with LDAP clients, with SRC applications, or with other applications through SRC APIs.

Table 26 shows the items for which you can define parameters and the methods you can use to define parameters for these items. See the documentation specified in the table for information about how to define parameters for each item.

Table 26: Parameter Definitions

Items That Contain Parameter Definitions	Methods by Which You Can Define the Parameter	Documentation That Describes How to Define Parameters
Global parameters, which you define in the runtime parameters in the policy repository folder.	SRC CLI Policy Editor SDX Admin	*Chapter 8, Overview of Using Local and Global Parameters*
Local parameters, which you define in policy groups.	SRC CLI Policy Editor	*Chapter 8, Overview of Using Local and Global Parameters*
System parameters, which are contained in the runtime parameters folder in the policy repository folder.	Subscriber sessions SRC network–values obtained when the SAE starts managing an interface	Table 24 on page 196
Value-added services in order of the precedence associated with the scopes associated with the service	LDAP client SDX Admin	*Setting Parameters for Value-Added Services* on page 53 *Restricting and Customizing Services for Subscribers* on page 82
Services in order of the precedence associated with the scopes associated with the service	SRC CLI	*Setting Parameter Values for Services with the SRC CLI* on page 8 *Restricting and Customizing Services for Subscribers* on page 27
Subscribers	SRC CLI LDAP client SDX Admin	*SRC-PE Subscribers and Subscriptions Guide, Chapter 12, Configuring Subscribers and Subscriptions with the SRC CLI* *SRC-PE Subscribers and Subscriptions Guide, Chapter 13, Configuring Subscribers and Subscriptions with SDX Admin*

Table 26: Parameter Definitions (continued)

Items That Contain Parameter Definitions	Methods by Which You Can Define the Parameter	Documentation That Describes How to Define Parameters
Subscriptions	SRC CLI SDX Admin Residential portal or enterprise service portal Dynamic Service Activator SAE's CORBA remote API	*SRC-PE Subscribers and Subscriptions Guide, Chapter 12, Configuring Subscribers and Subscriptions with the SRC CLI* *SRC-PE Subscribers and Subscriptions Guide, Chapter 13, Configuring Subscribers and Subscriptions with SDX Admin* *SRC-PE Sample Applications Guide* *SRC-PE Subscribers and Subscriptions Guide, Chapter 20, Overview of Enterprise Service Portals* *SRC Application Library Guide, Chapter 3, Activating Services with SOAP* SRC software distribution in the folder *SDK/doc/sae* or in the SAE CORBA remote API documentation on the Juniper Networks Web site at http://www.juniper.net/techpubs/software/management/sdx/api-index.html
Sessions	Residential portal Dynamic Service Activator SAE's CORBA remote interface	*SRC-PE Sample Applications Guide* *SRC Application Library Guide, Chapter 3, Activating Services with SOAP* SRC software distribution in the folder *SDK/doc/sae* or in the SAE CORBA remote API documentation on the Juniper Networks Web site at http://www.juniper.net/techpubs/software/management/sdx/api-index.html

Formatting Substitutions

Some SRC components handle the substitution syntax for you. For example, Policy Editor allows you to enter settings in fields, and it formats these settings in the correct syntax. In addition, IT managers or residential subscribers can enter settings through portals, and the portal formats these items in the correct syntax. You must enter some substitutions in SDX Admin using the correct syntax, however. Similarly, if you develop a portal that uses substitutions, you must use the correct syntax in the code for that portal.

A substitution has the following syntax:

```
[ ! ]<parameterName>[ :<role>]*=[<expression>]
[ //<comment> ]
```

- !—Fixes the substitution
- <parameterName>—Name of the parameter; either a name that you define or a name that is specified by the SRC software (see *Parameter Names* on page 403). If you are defining a substitution for a global parameter, make sure that the case of the parameter name in the substitution matches the case of the global parameter.
- <role>—Category of the parameter (see *Roles* on page 403)
- <expression>—A definition for the parameter (see *Expressions* on page 404)
- //<comment>—A comment about a substitution that appears on a new line after the substitution syntax (see *Adding Comments to Substitutions* on page 410)

Parameter Names

Parameter names identify the local or global parameters that are defined by you or that are specified by the SRC software. The parameter name is a string of alphanumeric characters starting with a letter that does not contain spaces or special characters. For more information about parameter names, see *Specifying Parameter Names* on page 405.

Roles

Parameters fall into different categories, known as roles in SDX Admin and types in the Policy Editor. For example, a parameter that defines an IP address has the role address. For more information about roles, see Table 26 on page 401.

Expressions

An expression in a parameter definition can take one the following values:

- An explicit value; for example, 1000000
- Another parameter; for example, a parameter called bodDestPort
- A mathematical expression that can include a combination of:
  - Parameters
  - Numbers—Integers and floating point numbers
  - Strings
  - IPv4 addresses
  - Ranges of numbers, strings, and addresses
  - Lists of values, such as lists of protocols
  - Maps—List of pairs of attributes and corresponding values
  - One keyword, **not**
  - Separators
  - Operators

  For example, `x == 1 ? rate : 2*rate`

The syntax for mathematical expressions is based primarily on Java syntax, although a few items use a proprietary syntax. When evaluating mathematical expressions, the SRC software:

- Follows a defined order for the precedence of operators (see *Using Operators* on page 407*)*.
- Performs all evaluations in long integer format until it finds an argument or result that is in Java floating point number format. Subsequently, the software performs evaluations in Java double floating point number format.
- Evaluates only subordinate expressions that meet the conditions for evaluation.
  - Evaluates only subordinate expressions that contain numbers and not parameters.
  - Stops the evaluation and substitutes the partial evaluation if an argument in double floating number format becomes an argument to an operator that takes only integers.
- Behaves in the same way as a Java evaluation if intermediate evaluations exceed or fall below the long integer range or the double floating point number range.

- Follows the Java rules for raising exceptions. For example, the software raises an exception if:
  - An evaluation involves a division by zero.
  - Literal numbers exceed the long integer limit or the double floating point number limit.

The following sections describe how to format the items that you can use in an expression.

Specifying Parameter Names

Observe the following rules when you are specifying parameter names:

- Enter a string of alphanumeric characters starting with a letter.
- Do not use spaces or special characters. For example, do not use the at sign (@) in a parameter name.
- You can use the underscore (_) and the dollar sign ($). Use the dollar sign to encode special characters by entering the Unicode equivalent of the character in hexadecimal format after the dollar sign. For example, use $0040 to encode the at sign (@).

Formatting Numbers

Observe the following rules when you are formatting numbers:

- Enter a digit after the decimal point in a floating point number. For example, you can use the number 4.0, but not the number 4.
- Do not enter characters that specify the type of number after that number. For example, do not enter the character L after a number to indicate that the number is a long integer.

Formatting Strings

Use Java syntax for strings; enclose strings in double quotation marks.

Example—“engineering”

Observe the following rules when you are formatting strings:

- Do not use octal escape sequences in strings. For example, do not use the escape sequence /137 in a string.
- Do not use Unicode escape sequences. For example, do not use the escape sequence \u80A6 in a string.

Using IPv4 Addresses

Use the following format for IP addresses:

<string>.<string>.<string>.<string> | '<string>.<string>.<string>.<string>'
<string> is a set of digits in the range 0–255

Example—'192.0.2.1'

Single quotation marks around an item indicate that it represents an address; however, for IPv4 addresses, the quotation marks are optional.

Specifying Ranges

To specify a range of numbers, strings, and addresses, use two dots between the arguments.

Example—192.0.2.1..192.0.3.1

Formatting Lists

To specify a list of values, enclose a set of subordinate expressions separated by commas in a pair of square brackets.

Example—[ip, icmp, ftp]

Formatting Maps

Maps are used to specify values that have optional and interdependent attributes. For example, when you define an application object through the Enterprise Manager portal, you can select a number of attributes and specify particular values for them. Depending on the value of the attribute, other attributes are possible or required.

To format a map, specify a list of pairs of attributes and corresponding values. Separate the pairs with commas, and enclose the list in curly brackets (braces).

Example—{applicationProtocol="ftp", sourcePort=123, inactivityTimeout=60}

Using Keywords

The SRC software ignores all Java keywords in substitutions, so that you can use Java keywords for identifiers such as variable names, function names, and attribute names in maps. The SRC software accepts one keyword, **not**, which is used to indicate conditions that do not match a specified value. For more information about the **not** keyword, see the *Using Operators* on page 407.

Using Separators

You cannot use a dot (.) as a separator. You can use other Java separators in the ways that Java supports.

Using Operators

Table 27 shows the operations and corresponding operators that the SRC software supports for substitutions. Most of the operators are Java operators, although a few operators are proprietary. You cannot use Java operators that do not appear in this table.

Table 27: Operations That You Can Use in Expressions

Operation	Operator	Number of Arguments	Result If Different from Java Conventions	Conditions for Evaluation	Example
Bitwise AND of the arguments	&	Two		Both arguments must be integers	234567 & 876543
Bitwise exclusive OR of the arguments	^	Two		Both arguments must be integers	234567 ^ 876543
Bitwise inclusive OR of the arguments	|	Two		Both arguments must be integers	234567 | 876543
Bitwise negation of the argument	~	One		Argument must be an integer	–234567
Difference between two arguments	–	Two		Both arguments must be numbers	876543 – 234567
Division of the first argument by the second argument	/	Two	Result of operation in double format	Both arguments must be numbers	589 / 756
Equal	= =	Two	Nonzero number if the arguments are equal	Both arguments must be numbers	rate = = 5
Greater than	>	Two	Nonzero integer if the first argument is greater than the second argument	Both arguments must be numbers	rate > 5
Greater than or equal to	> =	Two	Nonzero integer if the first argument is greater than or equal to the second argument	Both arguments must be numbers	rate > = 5
If... then... else... operation	?:	Three	If the first argument is nonzero, then the result is the second argument, else the result is the third argument	First argument must be a number	"x = = 1 ? rate : 2*rate"
Less than	<	Two	Nonzero integer if the first argument is less than the second argument	Both arguments must be numbers	rate < 5

Table 27: Operations That You Can Use in Expressions (continued)

Operation	Operator	Number of Arguments	Result If Different from Java Conventions	Conditions for Evaluation	Example
Less than or equal to	< =	Two	Nonzero integer if the first argument is less than or equal to the second argument	Both arguments must be numbers	rate < = 5
Logical AND	&&	Two	Nonzero integer if both the arguments are nonzero	Both arguments must be numbers	x = = 1 && y > = 5
Logical NOT	!()	One	Zero if the argument is nonzero	All arguments must be numbers	! x = = y
Logical OR	||	Two	Nonzero integer if at least one of the arguments is nonzero	Both arguments must be numbers	x = = 1 || y > = 5
Maximum of the arguments, max() = –infinity	max()	Zero or more		All arguments must be numbers	max (1, 3, 2, 4)
Minimum of the arguments, min() = +infinity	min()	Zero or more		All arguments must be numbers	min (1, 3, 2, 4)
Negation	–	One		Argument must be a number	–5
Not equal	! =	Two	Nonzero integer if the arguments are not equal	Both arguments must be numbers	rate ! = 5
Not match	not	One		None – expressions with this operator cannot be evaluated	not 192.0.2.1
Product of the arguments	*	Two		Both arguments must be numbers	rate*2
Raise the first argument to the power of the second argument	**	Two		Both arguments must be numbers	2**16
Range from the first argument to the second argument	..	Two		None—expressions with this operator cannot be evaluated	0..49
Remainder of division of the first argument by the second argument	%	Two		Both arguments must be integers	5%2
Round off the argument to the closest number	round()	One	Integer closest to the argument	Argument must be numbers	round(986532.654)
Round the argument down	floor()	One	Biggest integer less than or equal to the argument	Argument must be numbers	floor (986532.654)
Round the argument up	ceiling()	One	Smallest integer greater than or equal to the argument	Argument must be numbers	ceiling (986532.654)

Table 27: Operations That You Can Use in Expressions (continued)

Operation	Operator	Number of Arguments	Result If Different from Java Conventions	Conditions for Evaluation	Example
Shift the first argument left by the number of bits in the second argument	< <	Two		Both arguments must be integers	986532 < < 2
Shift the first argument right by the number of bits in the second argument	> >	Two		Both arguments must be integers	986532 > > 2
Sum of the arguments	+	One or two		Both arguments must be numbers	876 + 345 + 855

The precedence of the Java operators is the same as the precedence in Java; if you are unsure of the precedence of the operators, you can use parentheses to ensure that the software evaluates expressions in the desired way. For example, the following logical OR expression does not need parentheses.

```
x==1 || y>=5
```

You can, however, include parentheses as follows:

```
(x==1) || (y>=5)
```

The following list shows the precedence of the operators from lowest precedence to highest precedence:

- not
- ..
- ?:
- ||
- &&
- |
- ^
- &
- = =, ! =
- <, >, < =, > =
- < <, > >
- +, – (binary)
- *, /, %

- **
- +,- (unary)
- ~,!

Adding Comments to Substitutions

You can add a comment on the last line of the substitution. To do so:

1. Place the Java single-line comment marker (//) at the end of the last line of the substitution.
2. Enter the comment.

   There is no limit to the length of the comment you can enter. You do not need to use the new line marker in comments. Any text that follows the comment marker, regardless of how many lines the text spans, is treated as part of the comment.

Example—//This parameter specifies the QoS rate for this service.

The SRC software supports only the Java single-line comment marker. You cannot use the comment marker for multiple lines or comment markers for other languages.

Validating Substitutions

You can validate substitutions with Policy Editor, SDX Admin, and the Enterprise portal. For example, if you enter a substitution for a value-added service with SDX Admin, you can validate that substitution with SDX Admin.

When validating substitutions, the SRC software:

- Checks the syntax of substitutions. For example, if you incorrectly specify a range by using 3 dots between the arguments instead of 2 dots, the SRC software returns an error.
- Does not check the arguments that you specify for an operator. For example, in the expression 192.0.2.16/28 the software recognizes the forward slash (/) as a division operator, but does not check that the arguments are appropriate for division.

  This feature allows SRC components, such as the policy engine, to interpret the expression 192.0.2.16/28 as an IP address and mask rather than a division operation.

- Does not check for consistent use of roles in parameters in a chain of substitutions. For example, consider the following situation:

  1. You define in a policy group a local parameter x with the role network and an expression of y (x:network = y).

  2. You define in a service a parameter y with the role rate and a value of 123 (y:rate = 123).

     The software will substitute the value of 123 for x, even though 123 is a rate and not an address. Eventually, however, the substitution will cause problems, and a component such as the policy engine or the SAE will reject the value.

Example: Parameter Value Substitution

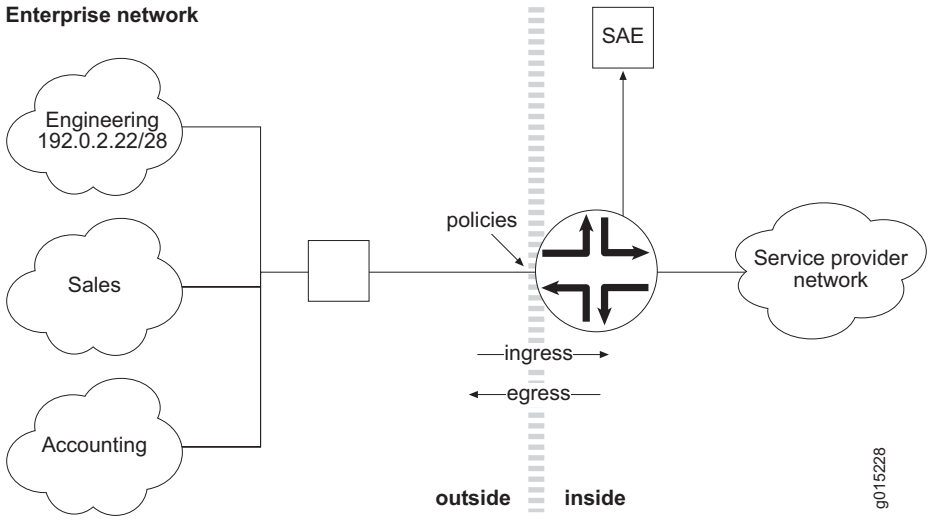
This section provides an example of how to use parameters and substitutions. It contains the following sections:

- Setting Up a Service That Uses Parameters on page 411
- Acquiring the Parameter Values on page 423

Setting Up a Service That Uses Parameters

In this example, we will create a value-added service that provides a gold-level quality of service. We will then subscribe this service to a department subnet in an enterprise network and be able to track and charge the department for the volume of bandwidth used. Figure 43 shows the network in our example.

Figure 43: Network Used in Parameter Substitution Example



From the service provider's perspective, the service provider's network is on the inside, and the enterprise network is on the outside. Ingress traffic flows from the enterprise network to the service provider's network. Egress traffic flows from the service provider's network to the enterprise network. The engineering department subnet in the enterprise network is the subnet that we will subscribe to the gold-level service and track.

The example uses two types of parameters (note that SDX Admin uses the term role in place of type):

- rate—Used to scale the rate limiter
- network—Used to specify IP subnets in classify conditions

Summary of Procedure

The following is a summary of the procedure we will use to set up the example.

1. Create a policy group called tierpolicy that classifies packets based on source and destination subnets and applies a rate limit action to those packets. The tierpolicy policy group contains three local parameters:
   - inside—Parameter of type network; used to specify a subnet
   - outside—Parameter of type network; used to specify a subnet
   - qos—Parameter of type rate; used to scale the rate limiter
2. Create a value-added service called GoldMetered, and assign tierpolicy as the policy group. In the GoldMetered service, configure the following parameter substitution:
   - qos—Fix to 50% of the interface_speed parameter. (interface_speed is a global runtime parameter that the SAE fills in with the actual speed of the router interface.)
   - dept—Create a parameter called dept that is parameter type (role) network.
   - outside—Set to dept (short for department), which effectively renames the outside parameter to dept.
   - inside—Set to any.
3. Create an enterprise subscriber, and configure the following parameter substitution:
   - eng—Create a parameter called eng (short for engineering department) that is parameter type (role) network, and set the value to 192.0.2.22/28.

4. Subscribe the subscriber to the GoldMetered service, and configure the following parameter substitution:

   - dept—Set to eng.

Creating a Policy Group

Use Policy Editor to create a policy group.

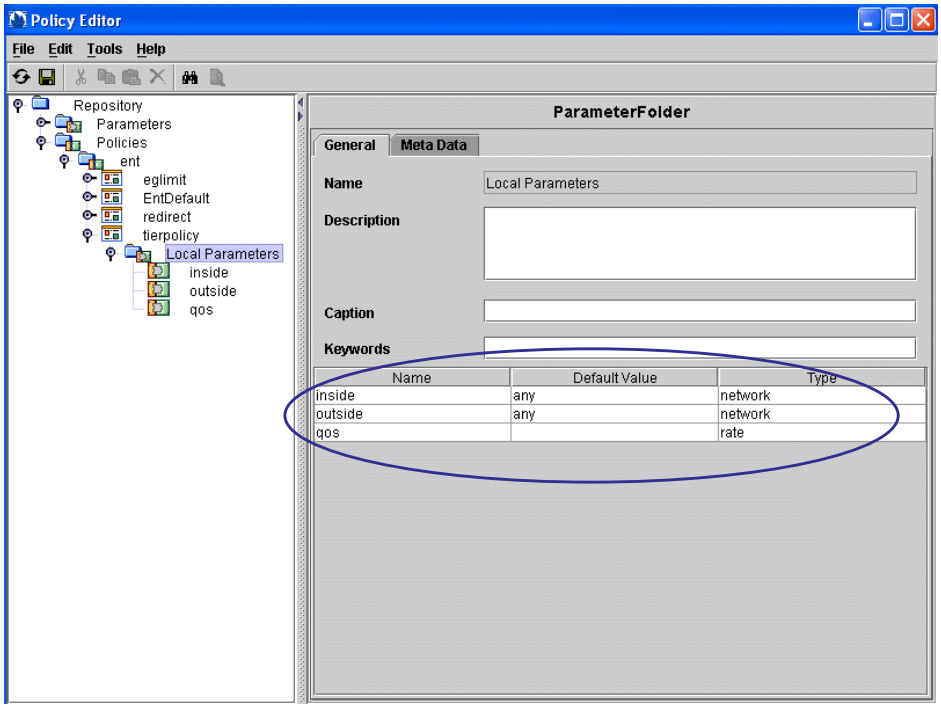
1. Create a policy group called tierpolicy.

2. Create the following local parameters, which are parameters that will be used only with tierpolicy.

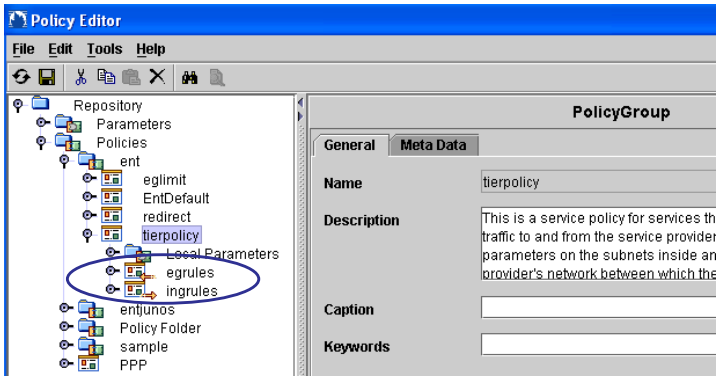
   - inside—Network parameter with a default value of any; any is a global parameter with value 0.0.0.0/0, which matches any network

   - outside—Network parameter with a default value of any; any is a global parameter with value 0.0.0.0/0, which matches any network

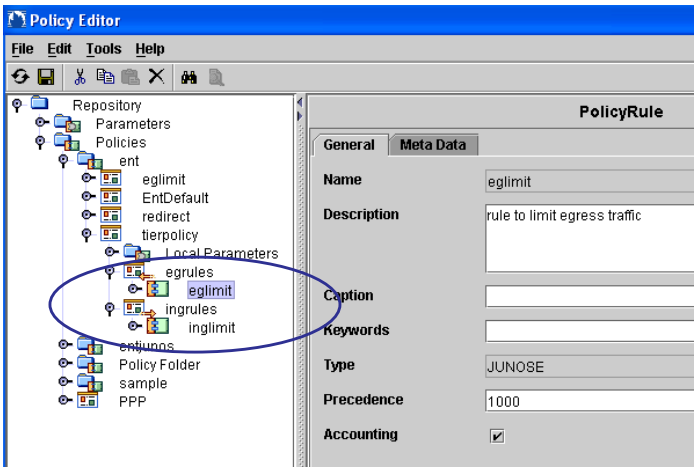
   - qos—Rate parameter



3. Create two policy lists, one for the ingress side of the interface, and one for the egress side of the interface.



4. Create two policy rules, one for ingress traffic and one for egress traffic.

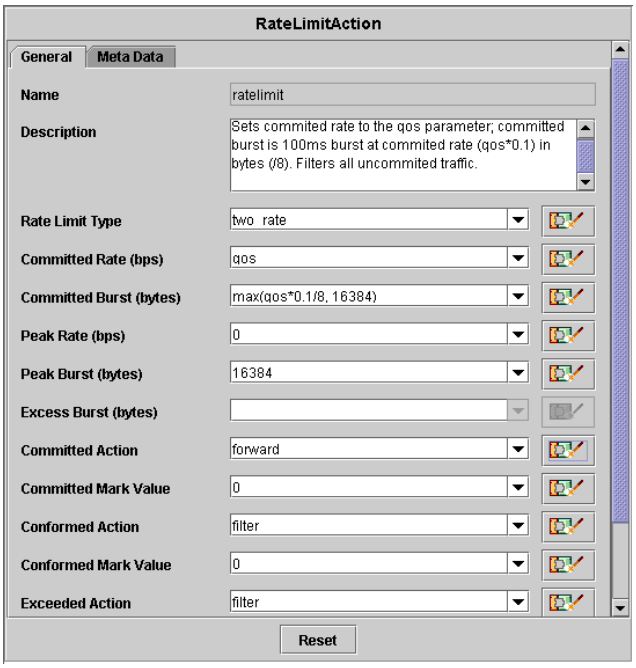


5. In the egress policy rule, which applies to traffic coming from the service provider network to the enterprise, create a condition that matches IP packets on source and destination networks:

   - source network = inside
   - destination network = outside

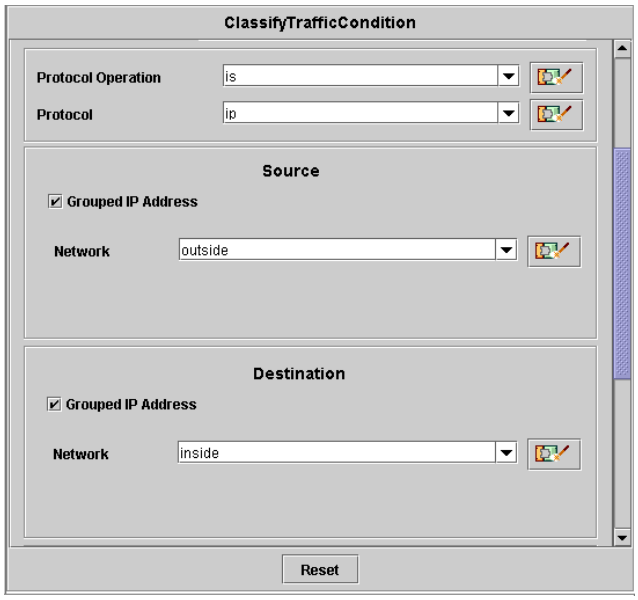
6. Also in the egress policy rule, create a rate-limit action that does the following:

   - Sets the committed rate to the qos parameter.
   - Sets the committed burst to the maximum of either 100 ms burst at committed rate (qos*0.1) in bytes (/8) or 16384.
   - Sets the peak burst to 16384.
   - Forwards all committed traffic.
   - Filters all uncommitted traffic.



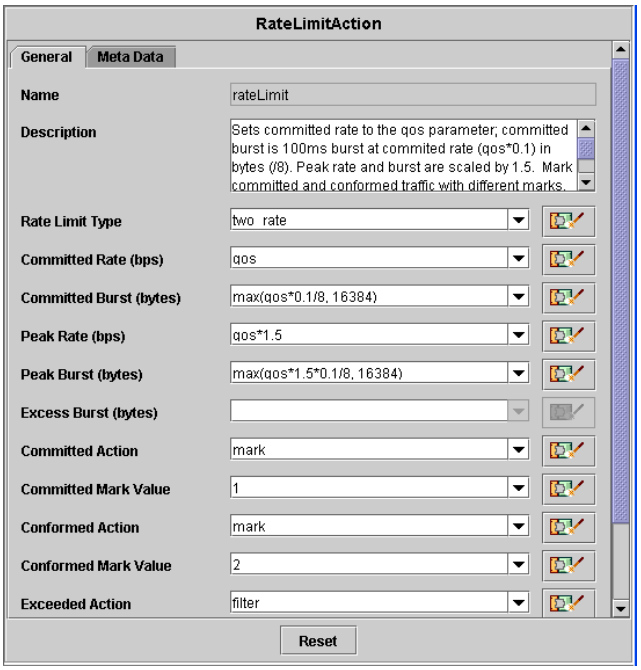
7. In the ingress policy rule, which applies to traffic coming from the enterprise network, create a condition that matches IP packets on source and destination networks:

   - source network = outside
   - destination network = inside

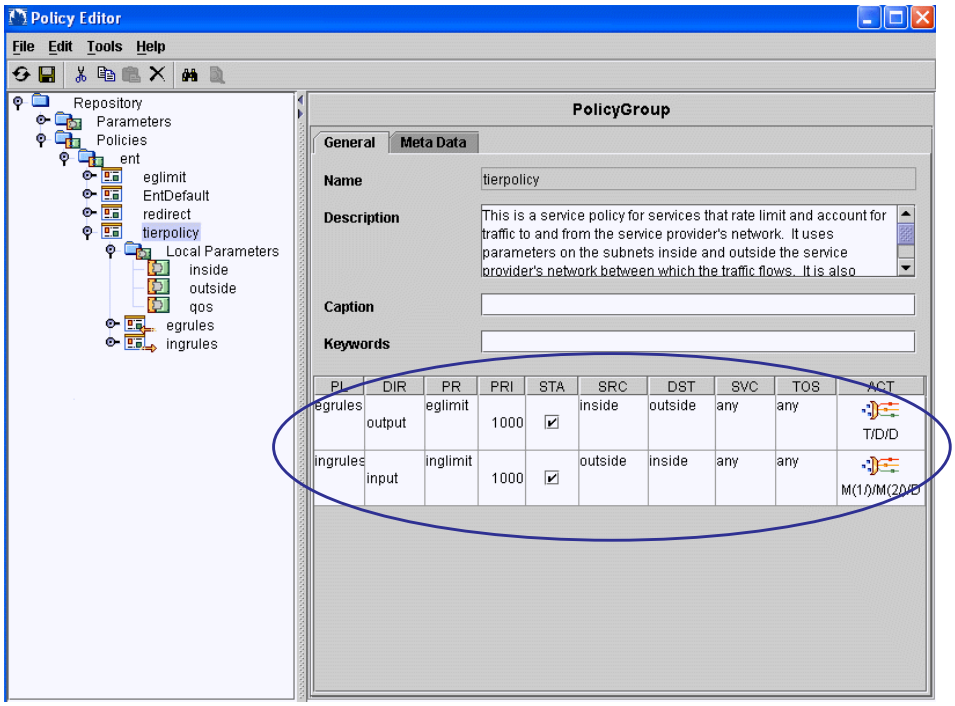


8. Also in the ingress policy rule, create a rate-limit action that does the following:

   - Sets the committed rate to the qos local parameter.
   - Sets the committed burst to the maximum of either 100 ms burst at the committed rate (qos*0.1) in bytes (/8) or 16384.
   - Scales the peak rate and burst by 1.5.
   - Marks committed and conformed traffic with different marks (1 and 2).
   - Drops all traffic that exceeds the rate limit.



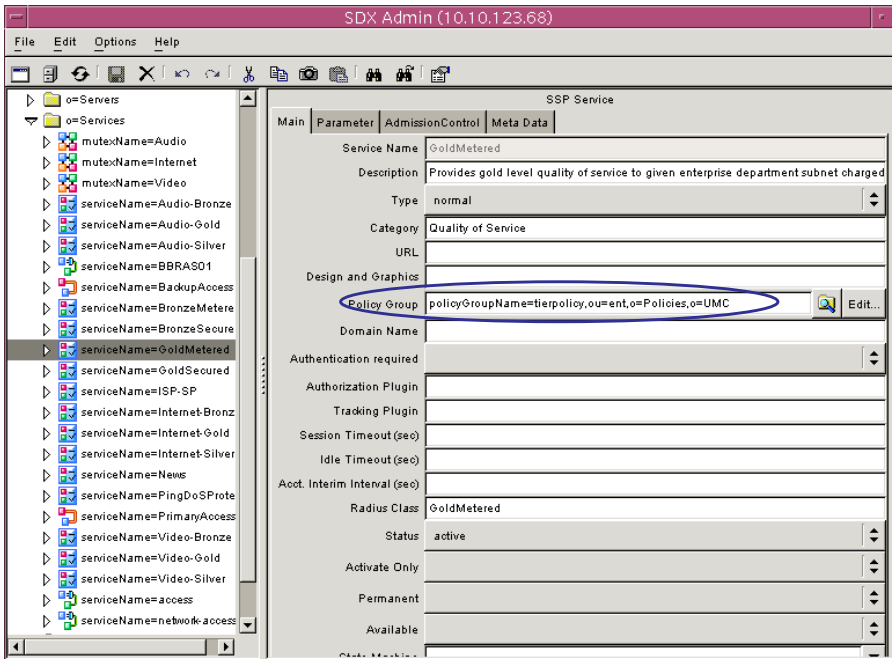
The policy group should now look like this:



Creating a Value-Added Service

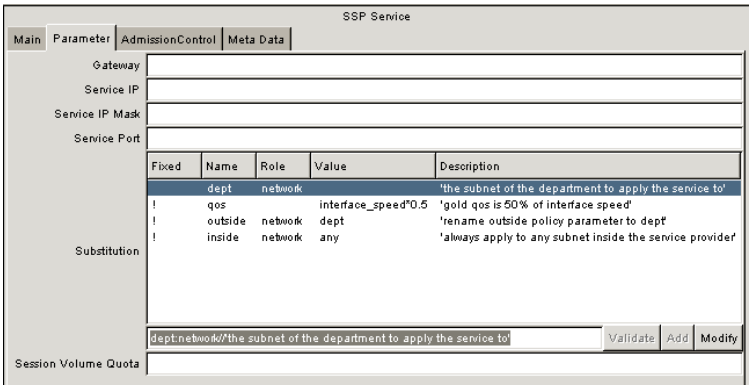
Use SDX Admin to create a value-added service.

1. Create a value-added service called GoldMetered, and assign tierpolicy as the policy group.



2. Select the **Parameter** tab of the GoldMetered service, and add the following parameters to the substitution table:

   - dept—Create a parameter called dept that is parameter type (role) network. This is the subnet of the department that the service will apply to.
   - qos—Fix the qos parameter to 50% of the interface_speed parameter. (interface_speed is a global runtime parameter that the SAE fills in with the actual speed of the router interface).
   - outside—Set the outside parameter to the value dept, which effectively renames the outside parameter to dept.
   - inside—Set the inside parameter to a value of any, which applies to any subnet inside the service provider's network.

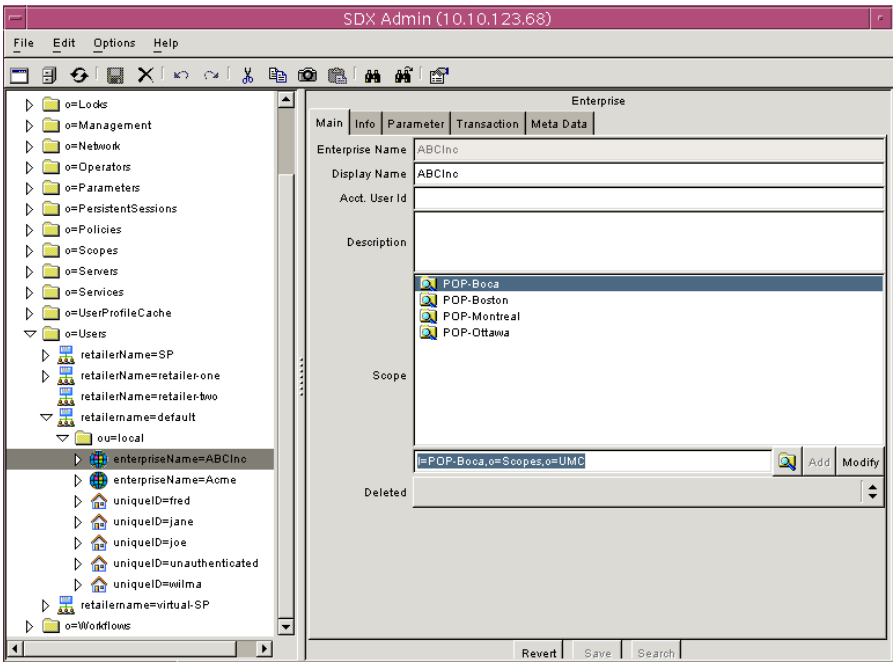


Creating an Enterprise Subscriber

The next step is to create an enterprise subscriber. Within the subscriber definition, create a parameter called eng that is parameter type (role) network, and set the value of eng to 192.0.2.22/28.

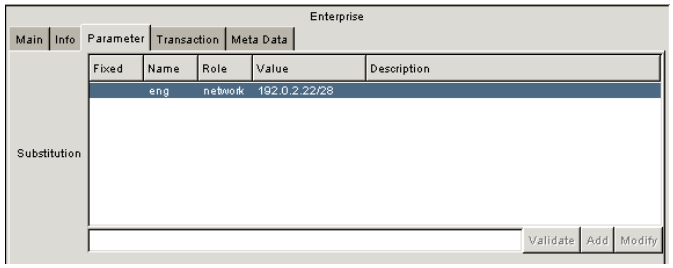
You create a subscriber by using SDX Admin or another directory client. You can create the eng parameter with SDX Admin or the sample enterprise service portal.

1. In SDX Admin, create an enterprise subscriber called ABCInc.

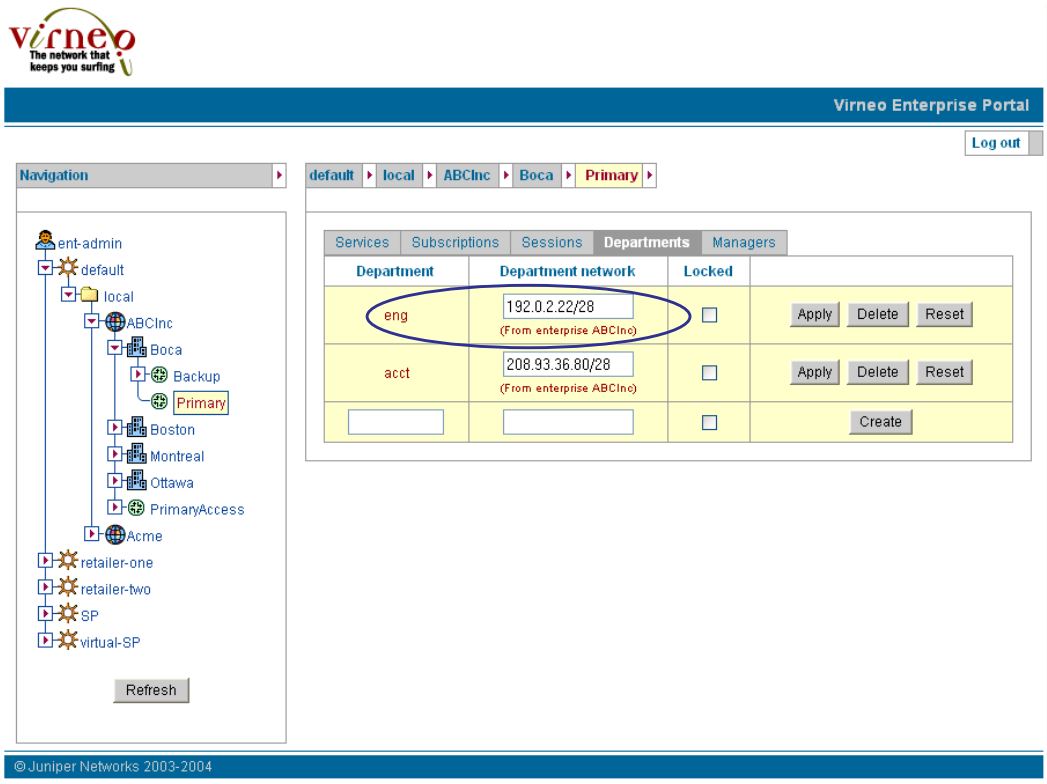


2. Create the eng parameter as part of the subscriber definition. You can perform this step by using either SDX Admin or the sample enterprise service portal.

   - To create the eng parameter in SDX Admin, select the **Parameter** tab of the ABCInc subscriber, and add the eng parameter to the substitution table.



   - To create the eng parameter in the sample enterprise service portal, select the **Departments** tab, add eng to the department field, and enter 192.0.2.22/28 as the network address of the department.



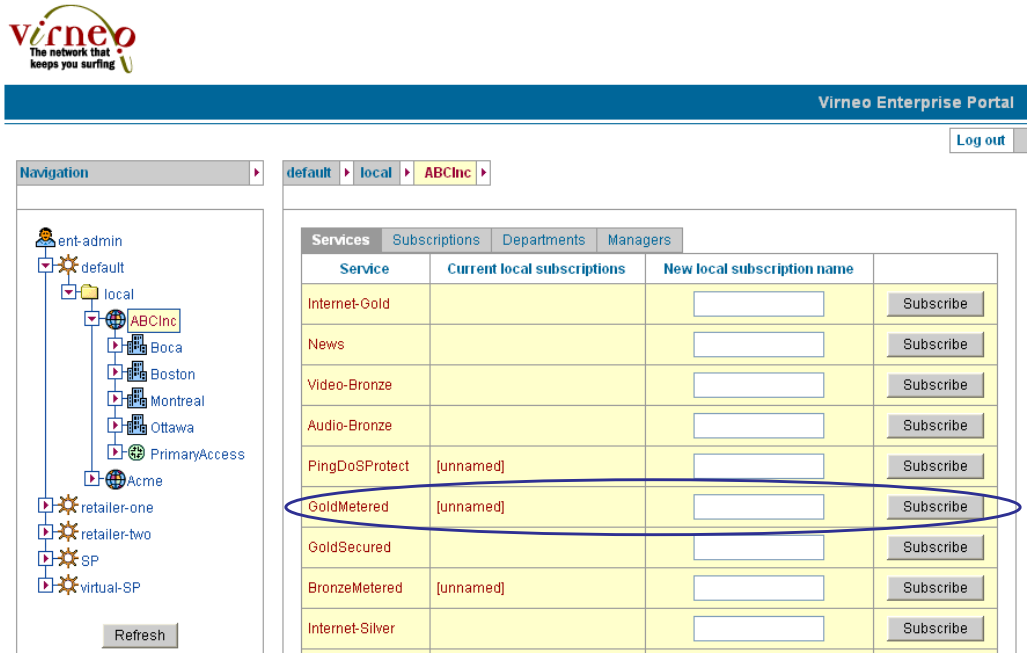
Subscribing ABCInc to the GoldMetered Service

Next, subscribe the ABCInc subscriber to the GoldMetered service. You can perform this step by using SDX Admin or the sample enterprise service portal.

In the sample enterprise service portal:

1. Select **ABCInc.** in the navigation pane.
2. Select the **Services** tab.

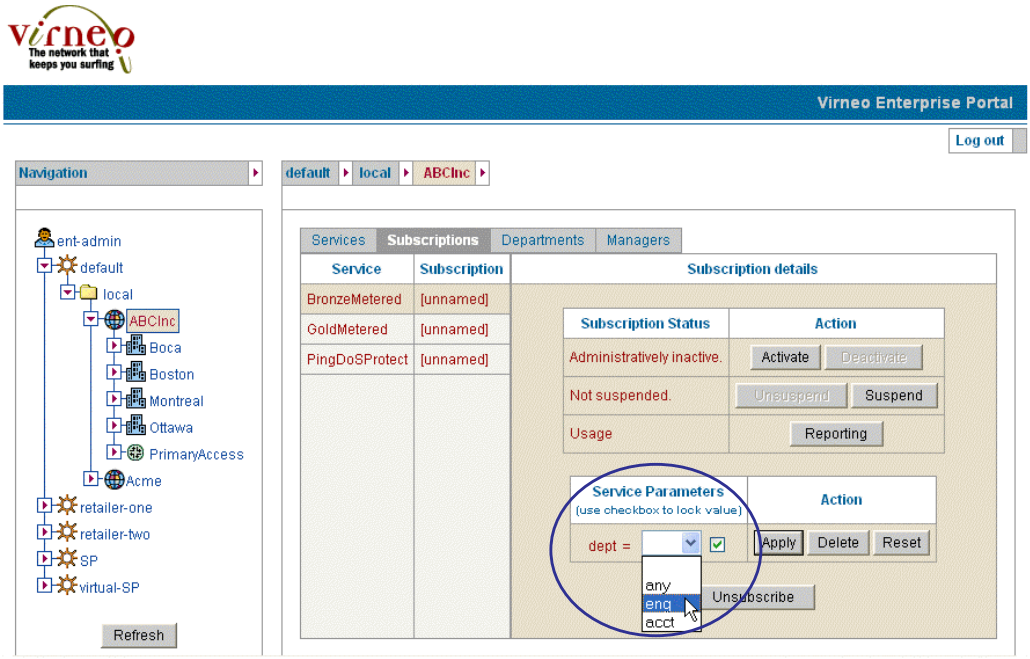
   The Services pane appears.



3. Click **Subscribe** in the GoldMetered service row.

4. Select the **Subscriptions** tab.

   The Subscriptions pane appears.



5. In the dept = field of the Service Parameters box, set the value of the dept parameter to eng.

Acquiring the Parameter Values

Once the SRC software has gone through the parameter value acquisition process, the three original parameters in the tierpolicy policy group have the following values:

- inside = 0.0.0.0/0

  This value was acquired from the global parameter any that was defined in the service definition.

- outside = 192.0.2.22/28

  This value was acquired as follows:

  - outside = dept—Acquired from the service definition
  - dept = eng—Acquired from the subscription
  - eng = 192.0.2.22/28—Acquired from the enterprise subscriber definition
  - qos = 500,000

This value was acquired from the service definition where the value of qos was set to 50 % of the interface_speed parameter. An interface_speed value of 1,000,000 was acquired from the router. If qos = 50 % of the interface speed, then the qos value is 500,000.

The rest of the rate-limit values are calculated based on the 500,000 value of qos.

Figure 44 shows the values of the ingress and egress policies that are applied to the router in our sample network.

Figure 44: Policies Applied to the Sample Network