

Chapter 16

Developing a Residential Portal

This chapter provides an overview of the SRC residential portal. The chapter contains the following sections:

- Before You Develop a Residential Portal on page 173
- Development Tools to Create a Residential Portal on page 174
- Virtual IP Address for Policies on page 175
- Redirecting Traffic to a Captive Portal Web Page on page 175
- Managing Security for Public Wireless LAN Applications on page 177
- Developing a Portal Based on the Sample Residential Portal on page 177

Before You Develop a Residential Portal

You can develop a residential portal based on the sample residential portal that accompanies the SRC software, or you can create a new one. Before you set up a residential portal, the SAE configuration for the retailers, services, subscribers, and basic subscriber services should already be in place.

Before you start to develop a portal, make sure that you understand the SAE configuration and how subscribers are expected to log in to the portal. See the following sources for information about the SAE and its configuration:

- *SRC-PE Network Guide, Chapter 1, Overview of the SAE*
- *SRC-PE Network Guide, Chapter 14, Configuring SRC Applications to Communicate with an SAE with the SRC CLI*

or

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

- *SRC-PE Subscribers and Subscriptions Guide, Chapter 3, Subscriber Logins and Service Activation*

When you are planning an SRC network that uses residential portals, consider how many instances of the portals you need. For example, if your network includes a number of different retailers, you could create different portals for different retailers. Residential portals use CORBA to connect to the SAEs, allowing you to create distributed Web applications. These applications can be deployed in clusters for load sharing.

Development Tools to Create a Residential Portal

The SRC software provides the following tools for service providers to make residential portals available to residential customers:

- CORBA remote API—Provides remote access to the SAE core API

The CORBA remote API is the preferred interface to use between external applications and the SRC software. See the following sources for more information:

- *SRC-PE Network Guide, Chapter 1, Overview of the SAE.*
- SAE CORBA remote API documentation in the SRC software distribution in the folder *SDK/doc/idl* or on the Juniper Networks Web site at

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

- Javadoc documentation for the sample residential portal—Provides information about the Java interface

You can access the Javadoc documentation for the sample portal from the Welcome page of the sample portal after you log in to the portal. See *Chapter 15, How Subscribers Use the Sample Residential Portal*.

- Sample residential portal

You can customize and extend the sample residential portal included with this release or create your own portal based on the sample. For information about the sample residential portal, see *Chapter 14, Installing and Configuring the Sample Residential Portal* and *Chapter 15, How Subscribers Use the Sample Residential Portal*.

Virtual IP Address for Policies

You can configure a virtual IP address to specify an IP address that policies use as a substitution to send traffic to a captive portal.

For information about how to configure a virtual IP address from the SRC CLI, see the documentation for the following statement in the *SRC-PE CLI Command Reference*:

```
shared sae configuration driver {
    virtual-portal-address virtual-portal-address;
}
```

Redirecting Traffic to a Captive Portal Web Page

A captive portal Web page is a page that receives redirected HTTP requests. You can use a captive portal page as the initial page a subscriber sees after logging in to a subscriber session and as a page used to receive and manage HTTP requests to unauthorized Web resources.

The type of information available from a captive portal page depends on the portal design. The page can provide informational messages or can let subscribers perform actions such as activating a service to which they have a subscription. For example, if a subscriber requests access to a service that the subscriber has not activated, the portal could display a captive portal page that tells the subscriber that the service is not available, or the page could prompt the subscriber to activate the requested service.

Implementing a captive portal requires the following:

- An instance of the redirect server installed on a host in the same network as a JUNOSe router. The redirect server redirects HTTP requests received from IP Filter to a captive portal page.
- When the SRC software is installed on a Solaris platform, the IP Filter tool installed and configured on the same host as the redirect server. This tool redirects incoming HTTP requests to the redirect server.
- Default policies installed on the JUNOSe router. The default policies on the JUNOSe router must include a forwarding or rate-limiting policy that permits access to the portal server and a next-hop rule to intercept the unauthorized access request packets. The target of the next-hop rule is the host on which the redirect server resides.
- A portal server for serving the captive portal pages.

For a sample captive portal, see the sample residential portal.

For information about configuring the redirect server, see the following chapters:

- *SRC-PE Subscribers and Subscriptions Guide, Chapter 15, Configuring Traffic Redirection with the SRC CLI*
- *SRC-PE Subscribers and Subscriptions Guide, Chapter 16, Configuring Traffic Redirection on a Solaris Platform*

Sequence for Redirecting Traffic

The following list describes the sequence of events that occurs when a subscriber tries to access a restricted service:

1. A subscriber opens a Web browser and attempts to access a restricted server; for example, `http://a.com`.
2. A next-hop policy on the JUNOS router sends this request to the redirect server instead of to the requested server.

The policy does not affect the destination address (resolved from `a.com`) in the IP packets.

3. For environments that have the SRC software installed on a Solaris platform, the IP Filter process running on the same host as the redirect server filters traffic and redirects traffic arriving on port 80 on the host's incoming interface.
4. The captured request is redirected to an address and a port where the redirect server listens.
5. The redirect server opens a TCP port (8800 by default) and sends the type of response configured—an HTTP 200 (OK) or a small HTML document that encodes a refresh in the meta header of the file—to the subscriber's browser for the requests.
6. The subscriber browser follows the redirect request and opens the captive portal page on the portal server.

Configuring the SRC Software in a Multihop Environment

The captive portal system implemented by the HTTP redirect server requires a single-hop connection; that is, the router accessed by the subscriber cannot be more than one hop away from the redirect server. However, some networking environments will require a multihop connection—through more than one router—to the redirect server.

You can use any of several methods to get around the intermediate, next-hop routers, such as IP-in-IP tunneling, deployment of a NAT device, and dynamic DNS. Contact Juniper Networks Professional Services for assistance with these methods.

Managing Security for Public Wireless LAN Applications

You can include in a residential portal a Web page that automatically refreshes itself and provides a keepalive application that verifies the HTTP session. If the keepalive application cannot verify the HTTP session, the portal terminates the subscriber session. This feature improves security for public wireless LAN applications.

If you include this Web page in a residential portal, the following sequence of events occurs:

1. When a subscriber logs in through the portal, the SRC software starts the keepalive application.
2. The keepalive application creates a session key and sends it to the residential portal.
3. The residential portal stores the session key in its corresponding HTTP session.
4. The keepalive application sets the timeout for the subscriber session to a value greater than the refresh time.
5. When the Web page refreshes itself, the keepalive application sends the session key to the residential portal.
6. The portal responds as follows:
 - If the session key matches the value in the portal's HTTP session, the portal updates the timeout for the subscriber session, creates a new session key, and sends the new key to the keepalive page.
 - If the session key does not match the value in the portal's HTTP session, the portal terminates the subscriber session.
7. If the Web page does not refresh itself before the timeout expires (for example, if the subscriber closes the Web browser or turns off the PC without logging out), the portal terminates the subscriber session.

Developing a Portal Based on the Sample Residential Portal

The source code is included with the sample residential portal. To modify the behavior of the portal beyond a simple configuration, install a Java development environment. You can find the source code of the sample residential portal in the directory *WEB-INF/src*. The portal pages are stored in the layout and tiles directories.

The sample residential portal does not require any specific environment, but the procedures below assume that you use the Eclipse platform. A servlet container is required to run the portals during development. We recommend that you use Tomcat and its Eclipse plug-in.

For information about your development environment, see the documentation for the product you are using.

Preparing to Develop a Portal Based on the Sample Residential Portal

The following instructions describe how to set up a development environment that uses Eclipse and Tomcat on a Solaris platform. If you want to use Eclipse and Tomcat on a different operating system, see the following Web sites:

- For Eclipse:

<http://www.eclipse.org>

- For Tomcat:

<http://jakarta.apache.org/tomcat>

To get ready to develop a portal based on the sample residential portal:

1. Download and install Eclipse from

<http://www.eclipse.org>

2. Download the Tomcat plug-in for Eclipse from

<http://www.sysdeo.com/eclipse/tomcatPlugin.html>

3. Unzip the plug-in into the Eclipse installation directory.

4. Download Tomcat from

<http://jakarta.apache.org/tomcat>

5. Install Tomcat:

```
mkdir $HOME/eclipse
cd $HOME/eclipse
unzip /tmp/eclipse-SDK-2.0.2-solaris-motif.zip
unzip /tmp/tomcatPluginV201.zip
cd $HOME
gzip -dc /tmp/tomcat-4.1.18.tar.gz | tar xvf -
```

6. Start Eclipse.

7. Configure the Tomcat plug-in.

Select **Window > Preferences > Tomcat**, and configure the Tomcat version and the path where you installed Tomcat.

Creating a Portal Project

To create a new Tomcat project inside Eclipse:

1. Select **File > New > Project > Java > Tomcat Project**, enter the name of the project, and click **Finish**.
2. Select **File > Import... > Zip File**, enter the path for *ssportal.war*; and click **Finish**.

3. Select **File > Properties > Java Build Path > Libraries > Add Jars**, open the sample project, navigate to *WEB-INF/lib*, and select all JAR files in the *WEB-INF/lib* directory.
4. Select **File > Properties > Tomcat**, and click **Can update server.xml file**.

Building the Portal

Eclipse automatically rebuilds the project when you save a modified source file.

To test or debug the project, run the code inside Tomcat.

To start Tomcat:

- Select **Tomcat > Start Tomcat**.

You can set break points in your code to debug the code.

Deploying the Portal

To create a new Web application, set the name of the target WAR file.

1. Select **File > Properties > Tomcat**.
2. Enter the path of the target WAR file in the field WAR file for export.
3. Right-click the portal project, and select **Tomcat Project > Export to the WAR file set** in project properties.
4. Copy the WAR file to the final deployment location; for example, */opt/UMC/jboss/server/default/deploy* on your portal server.

Testing a Portal Application

Simulated router drivers allow you to create subscriber sessions without connecting to a router. You can use a simulated router drive when you want to test your portal application. See *SRC-PE Monitoring and Troubleshooting Guide, Chapter 6, Configuring a Simulated Router Driver for Testing with the SRC CLI*.

