Sun GlassFish Web Space Server 10.0 Secure Web Access Add-On Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–7276
July 2009

090717@22510

Contents

◆ ◆ ◆ CHAPTER 1

Overview

The Secure Web Access Add-On for Sun GlassFish Web Space Server 10.0 software enables secure access of both Internet web sites and intranets.

Web Space Server offers browser-based secure remote access to portal content and services from any remote device. Web Space Server is a secure access solution that is accessible to users from any device with a Java™ technology-enabled browser, eliminating the need for a client software. Integration with Web Space Server ensures that users receive secure encrypted access to the content and services that they have permission to access.

Secure Web Access (SWA) software is targeted towards enterprises deploying highly secure remote access portals. These portals emphasize security, protection, and privacy of intranet resources. The architecture of Secure Web Access is well suited to these types of portals, which enable users to securely access intranet resources through the Internet without exposing the resources to the Internet.

This chapter includes the following topics:

Who Should Read This Guide?

This guide is intended for registered Web Space Server developers and administrators who want to use the Secure Web Access Add-On for Sun GlassFish Web Space Server package to enhance the power of Web Space Server software with secure remote login features.

For complete documentation for the core Sun GlassFish Web Space Server 10.0 software product, see the Sun GlassFish Web Space Server Document Collection. Some of the portal-related documentation is also available on OpenPortal web site.

Note – This guide does not provide detailed usage instructions for using Web Space Server in general. For such information, refer to the Sun GlassFish Web Space Server Document Collection. Also, this guide does not explain how to install and configure the Web Space Server. You need to have a working knowledge of how to install and configure the Web Space Server before installing the SWA Add-On.

What Are the Sun GlassFish Web Space Server Add-Ons?

The Sun GlassFish Web Space Server Add-Ons, also called *accelerators*, are an evolving set of stand-alone feature packages that provide performance enhancements or easier integration with third-party software tools. The Secure Web Access Add-On for the Sun GlassFish Web Space Server is one of the several add-on packages that are available in the Sun GlassFish Web Space Server software. See the Sun GlassFish Web Space Server product page for the most current list of add-on packages.

Each Web Space Server add-on package has its own user's guide. See the Sun GlassFish Web Space Server Add-On Document Collection for links to the documentation for the currently available add-on products. Check often, as the list of available add-ons is frequently updated.

Implementation

The high level design of Secure Web Access (SWA) provides access to intranet web contents from the Internet in a secure fashion. The two major components of SWA are Gateway and Rewriter. The Gateway component uses the Rewriter to rewrite the URLs in the contents that are obtained from the origin servers located in the intranet to point back to the Gateway.

SWA Gateway is implemented as a web application. You can off-load the low level networking and encryption to the web container or server to keep the Gateway code cleaner. The configuration of SSL and certificates will be more standard.

Gateway and Rewriter both use JCR as their data repository. The bundled JCR implementation is Jackrabbit, which is configured to use the local file system as the persistent data store for the ease of deployment out of the box. It can easily be reconfigured to use RDBMS or WebDAV in a production environment. Consult the Jackrabbit documentation for more details.

The SWA Gateway is a plain web application and the resources that it is trying to protect are all based on URLs of the intranet portal or non-portal web contents. Therefore, any access manager that is designed to protect web contents (such as OpenSSO, CA SiteMinder, IBM Tivoli Access Manager) can be used for authentication and access control. The SWA Gateway will be tested with the OpenSSO only out of the box.

Where Can You Get the Web Space Server Add-On Packages?

The Sun GlassFish Web Space Server add-on packages are available for free to registered Web Space Server users through the GlassFish Update Tool. The add-on packages that are available to you depend on how your Web Space Server software is registered:

- Registered users with a paid Web Space Server contract have unlimited access to the full set of Web Space Server add-ons.
- Registered users who do not have a paid Web Space Serverr service contract have access to a limited subset of the Web Space Server add-on collection.

Note – Although the Sun GlassFish Web Space Server software is a free, open-source product, the Web Space Server add-ons are proprietary components developed and licensed by Sun Microsystems, Inc.

To learn more about Web Space Server, add-on products for Web Space Server, and Web Space Server service contracts, refer to the Sun GlassFish Web Space Server product page.

Additional Sun GlassFish Web Space Server Documentation

Each Web Space Server Add-On package has its own user's guide. Please see the Sun GlassFish Web Space Server Add-On Document Collection for the links to documentation of the currently available add-on products. Please check the list often, as the list of available add-ons is frequently updated.

For complete documentation for the core Sun GlassFish Web Space Server 10.0 software product, see the Sun GlassFish Web Space Server Document Collection. Some of the portal-related documentation is also available on the OpenPortal website.

◆ ◆ ◆ CHAPTER 2

Getting and Installing Secure Web Access Add-On

This chapter explains how to download and install the Secure Web Access Add-On for Sun GlassFish Web Space Server software.

Before You Begin

This section explains the basic requirements and concepts you should review before proceeding to the installation of Secure Web Access Add-On for Web Space Server.

System Requirements

The Web Space Server software should be installed as described in the Chapter 2, "Web Space Server Installation Instructions," in *Sun GlassFish Web Space Server 10.0 Administration Guide*. Note that the requirements listed in the "Software and Hardware Requirements" in *Sun GlassFish Web Space Server 10.0 Installation Guide* also apply to the SWA Add-On.

If you are installing the SWA Add-On on a Web Space Server 10.0 update version, make sure that you are using a compatible version. For example, the Secure Web Access Add-On for Web Space Server 10.0 Update 5 should be installed only on a Web Space Server 10.0 Update 5 installation.

Installation Directories

The name of the core Web Space Server bundle is `webspace-`*version*`-for-glassfish.zip`), When you unzip this file, a `webspace-for-glassfish` directory is created. Throughout the installation instructions, this directory is referred to as *webspace-dir*.You can install and configure Web Space Server from this directory.

Similarly, the directory which contains the unzipped bundle of GlassFish is referred to as *glassfish-root*.

Platform-Specific Path Separators

The instructions and examples in this document use UNIX-style forward slash (/) path separators in file and command names. If Web Space Server and Sun GlassFish Enterprise Server are installed on a Windows system, be sure to use backslashes (\) instead of forward slashes. For example:

- **UNIX systems or Linux systems** - `glassfish/bin/asadmin`
- **Windows systems** - `glassfish\bin\asadmin`

Downloading the Secure Web Access Add-On

As with most Web Space Server add-on packages, the SWA Add-On is downloaded using the Sun GlassFish Update Tool.

Note – The version of Update Tool included with some versions of GlassFish Enterprise Server is not compatible with the Web Space Server add-on package repositories. You must use the version of Update Tool that comes with the Web Space Server 10.0 software.

Update Tool also includes a command-line (CLI) Image Packaging System (IPS) utility called `pkg`, which provides the same core functionality as its GUI-based counterpart.

▼ To Download the SWA Add-On Using the GUI-Based Update Tool

If you are running Update Tool for the first time, you will have to install the interface before proceeding.

1 **Start Update Tool by changing to the** *webspace-dir*`/bin` **directory and typing** `updatetool`**.**

- **If the Update Tool main window appears, proceed to Step 2.**
- **If a prompt appears asking whether to allow the installation of Update Tool to proceed:**

  a. **Type y to proceed.**

  The installer downloads and installs Update Tool and then exits. This process takes approximately 10 - 15 minutes.

  b. **Type the** `updatetool` **command again to start Update Tool.**

  The Update Tool main window appears.

2 **Expand Web Space under the Application Images pane, and choose Available Updates.**

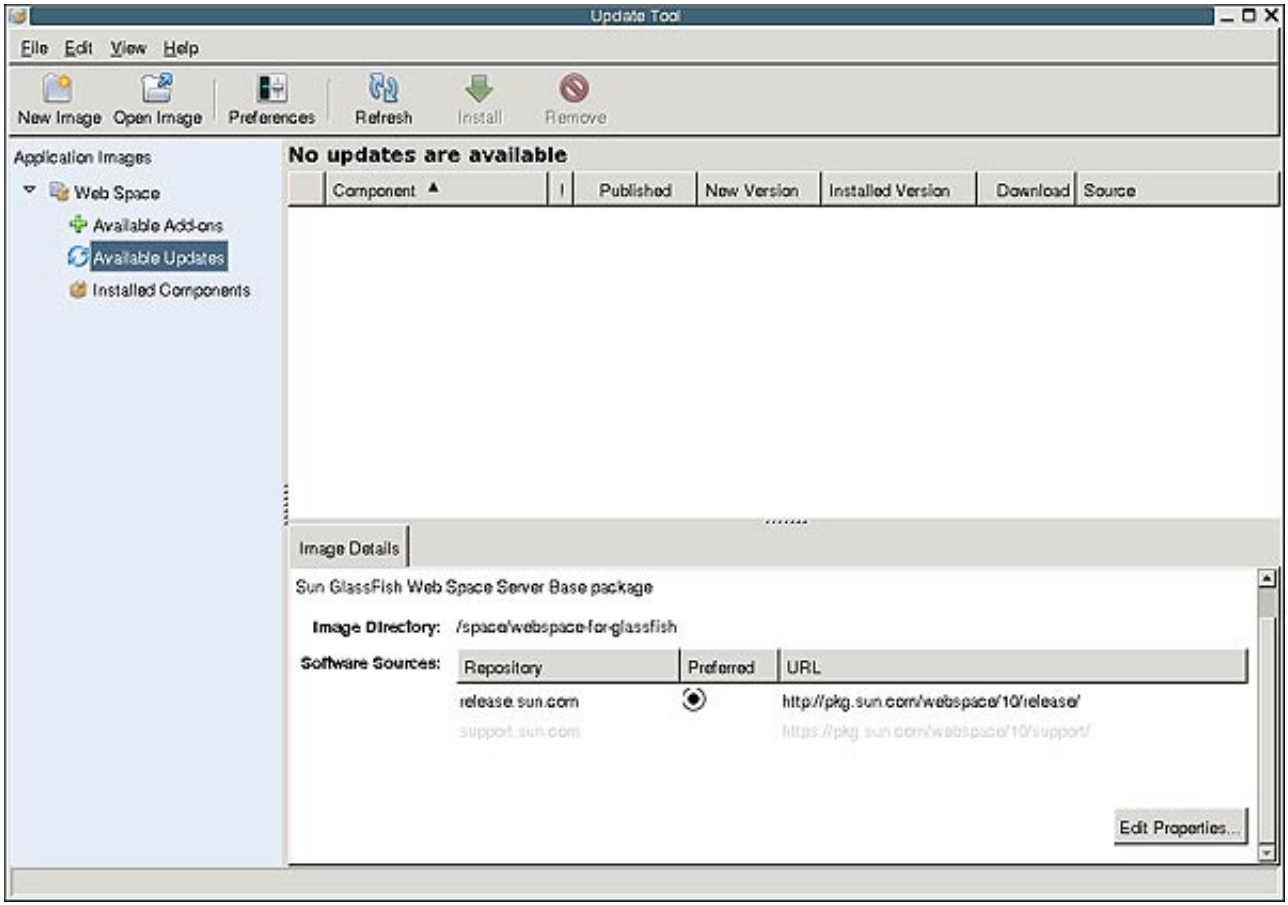

FIGURE 2–1 Update Tool

3 **Click the Edit Properties button.**

The Image Properties window is displayed.

4 **Select the option `support.sun.com` repository, and choose Preferred.**

The Repository Properties window appears.

5 **Provide the appropriate repository URL provided by Sun support, and click OK.**

The `support.sun.com` repository is highlighted.

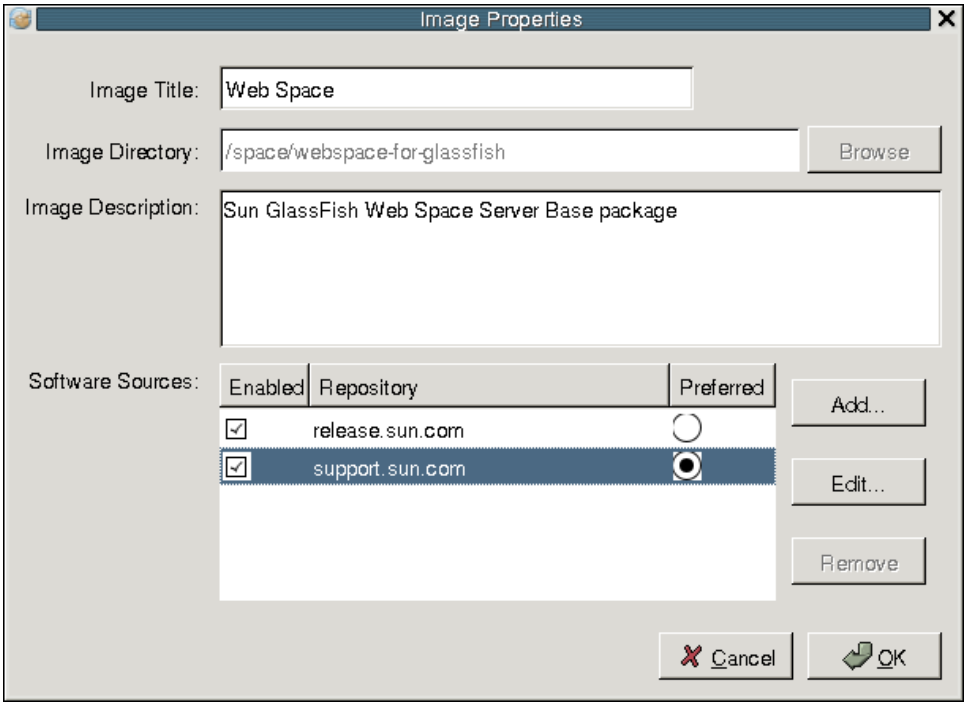

FIGURE 2–2 Selecting the `support.sun.com` repository

6. **Click OK again to enable the `support.sun.com` repository.**

   The Update Tool main window is displayed.

7. **Expand Web Space under the Application Images pane, and choose Available add-ons.**

8. **Select the Web Space for GlassFish V2 component, and click Install.**

9. **Select the `swa-administration`, `swa-datastore`, and `swa-gateway` components.**

Next Steps Proceed to "Installing the Secure Web Access Add-On" on page 13 for the remaining installation instructions.

▼ To Download the Secure Web Access Add-On Using the CLI-Based `pkg` Tool

1. **Start Update Tool by changing to the *webspace-dir*`/bin` directory and typing `updatetool`.**
   - **If the Update Tool main window appears, proceed to Step 2.**
   - **If a prompt appears asking whether to allow the installation of Update Tool to proceed:**
     a. **Type `y` to proceed.**
        The installer downloads and installs Update Tool and then exits. This process takes approximately 10 - 15 minutes.
     b. **Type the `updatetool` command again to start Update Tool.**
        The Update Tool main window appears.
2. **Change to the *webspace-dir*`/pkg/bin` directory.**
3. **Type the following command to download the Secure Web Access Add-On:**
   ```
   pkg set-authority -P --enable -O http://pkg.sun.com/webspace/10/ repository-name
   ```
   Ask your SunSolve service representative for the correct repository name to use.
4. **Type the following commands to install the base Secure Web Access Add-On.**
   ```
   pkg install webspace-swa-datastore-addon
   pkg install webspace-swa-administration-addon
   pkg install webspace-swa-gateway-addon
   ```

Next Steps Proceed to "Installing the Secure Web Access Add-On" on page 13 for the remaining installation instructions.

Installing the Secure Web Access Add-On

After using the Update Tool to get the SWA Add-On, the `swa-administration`, `swa-datastore`, and `swa-gateway` components are placed in the *webspace-dir*`/webspace` directory.

SWA uses JCR to store the Rewriter rule sets and other configuration data. The Gateway only reads from the data store, whereas the admin module both reads from and writes to the data store. Therefore, SWA has three components in terms of deployment. The Gateway is deployed

in the DMZ (demilitarized zone), and the admin module and the data store are usually deployed in the intranet. A DMZ is a small protected network between the public Internet and a private intranet, usually demarcated with a firewall on both ends.

In a simple deployment scenario, the three SWA components; `swa-datastore`, `swa-administration`, and `swa-gateway` are installed in the *webspace-dir*`/webspace` directory. Each component should be installed in its own subdirectory. Install the components in this order: `swa-datastore`, `swa-administration`, and `swa-gateway`.

▼ To Install the `swa-datastore` Component

1 **In a command shell of your operating system, change to the** *webspace-dir*`/webspace/swa-datastore` **directory.**

2 **Run** `ant -f install-glassfish.xml`**installation script..**

```
ant -f install-glassfish.xml
Buildfile: install-glassfish.xml

check-ant:

check-last-install:

set-last-install:

show-user-warning:
    [input] JAVA_HOME must be set to JDK 1.5 or greater and java must be available in the execution path. GlassFish mu


set-glassfish-properties:
    [input] Enter GlassFish Directory  [/opt/glassfish]
glassfish-root/glassfish
    [input] Enter GlassFish Domain (include full path to domain)  [glassfish-root/glassfish/domains/domain1]

    [input] Enter GlassFish Target  [server]

    [input] Enter GlassFish Administrator [admin]

    [input] Enter GlassFish Administrator Password File (include full path to file)  [/root/asadmin-password]

    [input] Enter GlassFish Administration Port  [4848]

    [input] Can installer deploy wars?  [true]


set-datastore-properties:
```

```
    [input] Enter Fully Qualified Datastore Host  [localhost]
***fully qualified name of the datastore deployment machine; can be the same machine***
    [input] Enter Datastore Port  [1099]

    [input] Use builtin Derby or a mySQL database  [Derby]
mysql

set-derby-properties:

set-mysql-properties:
    [input] Enter Database User Name  [root]
lportal
    [input] Enter Database User Password File (include full path to file)  [/root/database-user-password]

    [input] Enter Database Host  [localhost]

    [input] Enter Database Port  [3306]

    [input] Enter Database Name  [lportal]
```

Note – You need to provide the GlassFish root directory, path for the GlassFish administration password file and other details while installing the `swa-datastore` component of SWA.

You need to provide a FQDN. Apart from that, you need to specify the database, user name for the database, path for the database password file, and other attributes.

---

3 **Restart the GlassFish server.**

▼ To Install the `swa-administration` Component

1 **In a command shell of your operating system, change to the** *webspacedir*`/webspace/swa-administration` **directory.**

2 **Run** `ant -f install-glassfish.xml`**.**

```
ant -f install-glassfish.xml
Buildfile: install-glassfish.xml

check-ant:

check-last-install:

set-last-install:

show-user-warning:
```

```
    [input] JAVA_HOME must be set to JDK 1.5 or greater and java must be available in the execution path. GlassFish mu


set-glassfish-properties:
    [input] Enter GlassFish Directory  [/opt/glassfish]
glassfish-root/glassfish
    [input] Enter GlassFish Domain (include full path to domain)  [glassfish-root/glassfish/domains/domain1]

    [input] Enter GlassFish Target  [server]

    [input] Enter GlassFish Administrator [admin]

    [input] Enter GlassFish Administrator Password File (include full path to file)  [/root/asadmin-password]

    [input] Enter GlassFish Administration Port  [4848]

    [input] Can installer deploy wars?  [true]


set-datastore-properties:
    [input] Enter Fully Qualified Datastore Host  [localhost]
***fully qualified name of the datastore deployment machine; can be the same machine***
    [input] Enter Datastore Port  [1099]
```

Note – You need to provide the GlassFish root directory, path for the GlassFish administration password file and other details while installing the `swa-administration` component of SWA.

For the successful configuration of SWA, you need to provide the Fully Qualified Domain Name (FQDN). In a simple deployment scenario, where you are installing all the components on a single machine, FQDN can be your machine host name. You should not select the default `localhost`.

---

3 **Restart the GlassFish server.**

▼ To Install the `swa-gateway` Component

1 **In a command shell of your operating system, change to the** *webspace-dir*`/webspace/swa-gateway` **directory.**

2 **Run** `ant -f install-glassfish.xml`**.**

```
ant -f install-glassfish.xml
Buildfile: install-glassfish.xml

check-ant:
```

```
check-last-install:

set-last-install:

show-user-warning:
    [input] JAVA_HOME must be set to JDK 1.5 or greater and java must be available in the execution path. GlassF


set-glassfish-properties:
    [input] Enter GlassFish Directory  [/opt/glassfish]
glassfish-root/glassfish
    [input] Enter GlassFish Domain (include full path to domain)  [glassfish-root/glassfish/domains/domain1]

    [input] Enter GlassFish Target  [server]

    [input] Enter GlassFish Administrator [admin]

    [input] Enter GlassFish Administrator Password File (include full path to file)  [/root/asadmin-password]

    [input] Enter GlassFish Administration Port  [4848]

    [input] Can installer deploy wars?  [true]


set-datastore-properties:
    [input] Enter Fully Qualified Datastore Host  [localhost]
***fully qualified name of the datastore deployment machine; can be the same machine***
    [input] Enter Datastore Port  [1099]
```

Note – You need to provide the GlassFish root directory, path for the GlassFish administration password file and other details while installing the `swa-gateway` component of SWA.

Also, you need to provide the FQDN name.

◆ ◆ ◆ CHAPTER 3

SWA Administration

The Gateway Admin Portlet provides the user interface for SWA administration through the `SWA-datastore` component of the Secure Web Access Add-On. After installing the `SWA-administration` component, you can access Gateway Admin Portlet can be accessed from the control panel of Web Space Server.

Gateway Admin Portlet

The Gateway Admin Portlet has the following tabs:

Rewriter Tab

Use this tab to configure the Rewriter settings.

Rewriting of All URIs

If this option is enabled, all URLs are rewritten. If this option is disabled, only intranet URLs are rewritten. Intranet URLs are the URLs with hosts that are in the domains or subdomains listed in the Proxies for Domains and Subdomains list under the Proxy tab. This option is disabled by default.

URIs Not to Rewrite

This list indicates URLs that will not be rewritten even when *Rewriting of All URIs* is enabled. Use the Add Row and Delete Row buttons to add URLs to the list and remove URLs from this list.

Map URIs to RuleSets

Indicates which RuleSet to apply for a particular URL. You can use the Add Row button to add an URI and a RuleSet for it. Similarly, you can use the Delete Row button to delete a row corresponding to a URI and a RuleSet. The default Rulesets are, `inotes_ruleset`, `exchange_2000sp3_owa_ruleset`, `exchange_2003_owa_ruleset`, `sap_portal_ruleset`, `iplanet_mail_ruleset`, and `default_gateway_ruleset`.

Map Parser to MIME Types

Indicates which parser to use for a particular MIME type. You can use the Add Row and Delete Row tabs to add and delete rows for a parser and its associated MIME types.

Map Parser to URIs

Indicates which parser to use for a resource with a particular extension. You can use the Add Row and Delete Row tabs to add and delete rows for a parser and its associated URIs.

RuleSet Tab

Use this tab to manage the Rewriter rulesets. For information on how to write a Rewriter ruleset, see Chapter 4, "Working with Rewriter."

Proxy Tab

Use this tab to configure the proxy settings of the Gateway.

Use Proxy

Use a proxy server to connect to origin servers if this flag is enabled. This option is disabled by default.

Proxies for Domains and Subdomains

The proxy server to use for an origin server in the specified domain and subdomain. If a proxy is not specified for a given domain and subdomain, or if the Use Proxy option is disabled, a direct connection will be made. The entries in this list are also used by the Rewriter. The Rewriter rewrites all URLs whose domains match the domains listed in this list.

Proxy Password List

Specify the user name and password required for the Gateway to authenticate to a specified proxy server, if the proxy server requires authentication to access some or all the sites.

Miscellaneous Tab

Use this tab to configure the miscellaneous Gateway settings.

Default Domain and Subdomain

The default domains are useful when URLs contain only the host names without the domain and subdomain. In this case, the Gateway assumes that the host names are in the default domain list and proceeds accordingly.

Default Landing URL

This field indicates the URL of the destination page to show if the Gateway is accessed without a destination being specified. If this attribute is not set, a page with a text field is shown.

Cookies That Are Forwarded Unchanged

The cookies set by origin servers are usually transformed to act as if they were set by the Gateway when sending the response to the web clients. The cookies specified in this list are forwarded to the web clients unchanged. This option is useful for SSO cookies when a policy agent used to protect the Gateway is expected from the SSO server.

About the Gateway

The Gateway provides the interface and security barrier between remote user sessions originating from the Internet and your corporate intranet. The Gateway securely presents content from internal web servers and application servers through a single interface to a remote user.

The Gateway resides in the demilitarized zone (DMZ). The Gateway provides a single secure access point to all intranet URLs and applications, thus reducing the number of ports to be opened in the firewall. All other Web Space Server services reside behind the DMZ in the secured intranet. Communication from the client browser to the Gateway is encrypted using HTTP over Secure Sockets Layer (SSL). Communication from the Gateway to the server and intranet resources can be either HTTP or HTTPS.

In Secure Mode, SSL is used to encrypt the connection between the client and the Gateway over the Internet. SSL can also be used to encrypt the connection between the Gateway and the server. The presence of the Gateway between the intranet and the Internet extends the secure path between the client and the Web Space Server.

The `swa-gateway` component of the Secure Web Access Add-On holds the Gateway functionality. After you install the gateway component, you can access Web Space Server through the Gateway. Consider a simple deployment where you are installing Web Space Server

and the three SWA components on a single machine. If `http://`*machine-name*`:8080` is the URL for the local instance of Web Space Server, you can send HTTP and HTTPS requests to access the gateway using the URL `http://`*machine-name*`:8080/gateway/index.jsp`. You can type a URL in the *Enter the URL you want to access:* box and click Go to access a web application. In this scenario, type `http://`*machine-name*`:8080` to be redirected to your Web Space Server instance.

For the detailed instructions about how to use a Gateway to access an OpenSSO or Access Manager Policy Agent, see Chapter 5, "Policy Agent."

◆ ◆ ◆ CHAPTER 4

Working with Rewriter

The Rewriter component of Secure Remote Access enables users to access intranet web pages through the Gateway by parsing the web pages.

This Chapter covers the following topics:

Introduction to Rewriter

The Rewriter component of Secure Remote Access enables end users to browse the intranet by modifying Uniform Resource Identifier (URI) references on web pages so that they point to the Gateway. A URI defines a way to encapsulate a name in any registered name space, and labels it with the name space. The most common kinds of URIs are Uniform Resource Locators (URLs). Rewriter supports only HTTP or HTTPS. This support is regardless of the capitalization of the protocol. Rewriter only supports backslashes when they are part of a relative URL.

EXAMPLE 4–1 Rewriting URLs

`http://abc.sesta.com\\index.html` is rewritten.

These URLs are not rewritten: `http:\\\\abc.sesta.com`. *http:/abc.com*

Character Set Encoding

HTTP standards require that HTTP headers or HTML meta tags specify a character set for web pages. However, sometimes this information is not available. The character set must be known so that encoding for the data is set and the data is displayed as intended by the creator.

To detect the character sets, install the `SUNWjchdt` package from the Java Enterprise System Accessory CD. If this product is installed, Rewriter will detect it and use it if necessary.

Note – Using this product can affect performance, so you should install it only when required. See the `jcharset_readme.txt` for details on installation, configuration and usage.

Rewriter Usage Scenarios

When a user tries to access intranet web pages through the Gateway, web pages are made available by using Rewriter. Rewriter is used by the URL Scraper and the Gateway.

URL Scraper

The URL Scraper provider gets content from configured URIs. Before sending these URIs to the browser, it expands all relative URIs to absolute URIs.

For example, if a user is trying to access a site as:

```
<a href="../mypage.html">
```

Rewriter translates this to:

```
<a href="http://yahoo.com/mypage.html">
```

where `http://yahoo.com/test/` is the base URL of the page.

See the *Sun Java SystemPortal Server Administration Guide* for details about the URLScraper provider.

Gateway

The Gateway obtains content from Internet portals. Before sending the content to the browser, it prefixes the Gateway URI to the existing URI so that subsequent URI requests from the browser can reach the Gateway.

For example, a user who is trying to access an HTML page on an Internet machine as:

```
<a href="http://mymachine.intranet.com/mypage.html>"
```

Rewriter prefixes this URL with a reference to the Gateway as follows:

```
<a href="https://gateway.company.com/http://mymachine.intranet.com/ mypage.html>"
```

When the user clicks a link associated with this anchor, the browser contacts the Gateway. The Gateway fetches the content of `mypage.html` from `mymachine.intranet.com`.

The Gateway uses several rules to determine the elements of a fetched web page that will be rewritten.

Writing Rulesets

For more information about defining a ruleset, see the *Portal Server Administration Guide*. After creating a new ruleset, you need to define the required rules.

This section covers the following topics:

Public Interface (RuleSet DTD)

RuleSet DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  CDDL HEADER START
  The contents of this file are subject to the terms
  of the Common Development and Distribution License
  (the License). You may not use this file except in
  compliance with the License.

  You can obtain a copy of the License at
  http://www.sun.com/cddl/cddl.html and legal/CDDLv1.0.txt
  See the License for the specific language governing
  permission and limitations under the License.
```

```
  When distributing Covered Code, include this CDDL
  Header Notice in each file and include the License file
  at legal/CDDLv1.0.txt.
  If applicable, add the following below the CDDL Header,
  with the fields enclosed by brackets [] replaced by
  your own identifying information:
  "Portions Copyrighted [year] [name of copyright owner]"

  Copyright 2008 Sun Microsystems Inc. All Rights Reserved
  CDDL HEADER END
-->

<!--
    The following constraints are not represented in DTD, but taken care
    programatically
    1. In a Rule, All Mandatory attributes cannot be "*".
    2. Only one instance of the below elements is allowed, but in any order.
    1)HTMLRules
    2)JSRules
    3)XMLRules
    3. ID should alway be in lower case.
-->
<!ENTITY % gtype 'GROUPED'>
<!ENTITY % stype 'SCATTERED'>

<!ENTITY % eURL 'URL'>
<!ENTITY % eEXPRESSION 'EXPRESSION'>
<!ENTITY % eDHTML 'DHTML'>
<!ENTITY % eDJS 'DJS'>
<!ENTITY % eSYSTEM 'SYSTEM'>

<!ENTITY % ruleSetElements '(HTMLRules | JSRules | XMLRules)?'>
<!ENTITY % htmlElements '(Form | Applet | Attribute | JSToken)*'>
<!ENTITY % jsElements '(Variable | Function)*'>
<!ENTITY % xmlElements '(Attribute | TagText)*'>

<!ELEMENT RuleSet (%ruleSetElements;,%ruleSetElements;,%ruleSetElements;)>
<!ATTLIST RuleSet
    type (%gtype; | %stype;) "GROUPED"
    id ID #REQUIRED
    extends CDATA "none"
>

<!ELEMENT HTMLRules (%htmlElements;)>
<!ATTLIST HTMLRules
    type (%gtype; | %stype;) "GROUPED"
    id CDATA "html_rules"
>
```

```
<!ELEMENT Form EMPTY>
<!ATTLIST Form
    name CDATA #REQUIRED
    field CDATA #REQUIRED
    valuePatterns CDATA ""
    source CDATA "*"
>

<!ELEMENT Applet EMPTY>
<!ATTLIST Applet
    code CDATA #REQUIRED
    param CDATA "*"
    valuePatterns CDATA ""
    source CDATA "*"
>
<!ELEMENT JSToken (#PCDATA)>


<!ELEMENT JSRules (%jsElements;)>
<!ATTLIST JSRules
    type (%gtype; | %stype;) "GROUPED"
    id CDATA "js_rules"
>

<!ELEMENT Variable (#PCDATA)>
<!ATTLIST Variable
    name CDATA ""
    type (%eURL; | %eEXPRESSION; | %eDHTML; | %eDJS; | %eSYSTEM;) "EXPRESSION"
    source CDATA "*"
>

<!ELEMENT Function EMPTY>
<!ATTLIST Function
    name CDATA #REQUIRED
    paramPatterns CDATA #REQUIRED
    type (%eURL; | %eEXPRESSION; | %eDHTML; | %eDJS;) "EXPRESSION"
    source CDATA "*"
>


<!ELEMENT XMLRules (%xmlElements;)>
<!ATTLIST XMLRules
    type (%gtype; | %stype;) "GROUPED"
    id CDATA "xml_rules"
>
<!ELEMENT TagText EMPTY>
<!ATTLIST TagText
    tag CDATA #REQUIRED
```

```
    attributePatterns CDATA ""
    source CDATA "*"
>
<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute
    name CDATA #REQUIRED
    tag CDATA "*"
    valuePatterns CDATA ""
    type (%eURL; | %eDHTML; | %eDJS; ) "URL"
    source CDATA "*"
>
```

Note – You can use * as a part of rule value except that mandatory attribute values cannot be just *. Such rules are ignored, but the message is logged in the RuleSetInfo log file. For information on this log file, see "Debug File Names" on page 56.

Sample XML DTD

This section contains a sample rule set. The "Case Study," on page 140 is used to illustrate how these rules are interpreted by Rewriter.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
Rules for integrating a mail client with the gateway.
-->
<!DOCTYPE RuleSet SYSTEM "jar://rewriter.jar/resources/RuleSet.dtd">
<RuleSet type="GROUPED" id="owa">
<HTMLRules>
<Attribute name="action" />
<Attribute name="background" />
<Attribute name="codebase" />
<Attribute name="href" />
<Attribute name="src" />
<Attribute name="lowsrc" />
<Attribute name="imagePath" />
<Attribute name="viewClass" />
<Attribute name="emptyURL" />
<Attribute name="draftsURL" />
<Attribute name="folderURL" />
<Attribute name="prevMonthImage" />
<Attribute name="nextMonthImage" />
<Attribute name="style" />
<Attribute name="content" tag="meta" />
</HTMLRules>
<JSRules>
<!-- Rules for Rewriting JavaScript variables in URLs -->
<Variable name="URL"> _fr.location </Variable>
```

```
<Variable name="URL"> g_szUserBase </Variable>
<Variable name="URL"> g_szPublicFolderUrl </Variable>
<Variable name="URL"> g_szExWebDir </Variable>
<Variable name="URL"> g_szViewClassURL </Variable>
<Variable name="URL"> g_szVirtualRoot </Variable>
<Variable name="URL"> g_szBaseURL </Variable>
<Variable name="URL"> g_szURL </Variable>
<Function name="EXPRESSION" name="NavigateTo" paramPatterns="y"/>
</JSRules>
<XMLRules>
<Attribute name="xmlns"/>
<Attribute name="href" tag="a"/>
<TagText tag="baseroot" />
<TagText tag="prop2" />
<TagText tag="prop1" />
<TagText tag="img" />
<TagText tag="xsl:attribute"
attributePatterns="name=src" />
</XMLRules>
</RuleSet>
```

Procedure to Write Rules

The general procedure to write rules is:

- Identify the directories that contain the HTML pages whose content needs to be rewritten.
- In these directories, identify the pages that need to be rewritten.
- Identify the URLs that need to be rewritten on each page. An easy way identify most of the URLs is to search for "http" and "/".
- Identify the content type of the URL: HTML, JavaScript or XML.
- Write the rule required to rewrite each of these URLs by editing the required ruleset in the Rewriter service under Portal Server Configuration in the Access Manager administration console.
- Combine all the rules into a ruleset for that domain.

Ruleset Guidelines

When creating a ruleset, keep the following in mind:

- The order of precedence for specific hosts is based on the longest URI match. For example for the following rulesets

  ```
  mail1.central.abc.com|iplanet_mail_ruleset
  *.sfbay.abc.com|sfbay_ruleset
  *.abc.com|generic_ruleset
  ```

  `sfbay_ruleset` is used as it has the longest match.

- The rules in the ruleset are applied to each statement in the page until a rule matches a particular statement.

  While writing the rules, keep in mind the order of the rules. Rules are applied to the statements in a page, in the order in which they occur in the ruleset. If you have specific rules, and general rules that contain a "*", define the specific rules first, then the general rules. Otherwise, the general rule is applied to all statements, even before the specific rule is encountered.
- All rules need to be enclosed within the `<RuleSet>` `</RuleSet>` tags.
- Include all rules that need to rewrite HTML content in the `<HTMLRules>` `</HTMLRules>` section of the ruleset.
- Include all rules that need to rewrite JavaScript content in the `<JSRules>` `</JSRules>` section of the ruleset.
- Include all rules that need to rewrite XML content in the `<XMLRules>` `</XMLRules>` section of the ruleset.
- In your intranet pages, identify the URLs that need to be rewritten, and include the required rules in the appropriate sections (HTML, JSRules, or XMLRules) of the ruleset.
- Assign the ruleset to the required domain.
- Restart the Gateway to affect any changes:

  gateway-install-root`/SUNWportal/bin/gateway -n` *gateway-profile-name* `start`

Defining the RuleSet Root Element

The ruleset root element has two attributes:

- `RuleSetName`, for example, `default_ruleset.` This name is referenced in RuleSet to URI mapping.
- `Extends`. This attribute refers to the inheritance feature of rulesets. The value points to the ruleset from which you would like to derive a ruleset.

  Use the value `none` to signify that this new, independent ruleset does not depend on any other ruleset, or specify the <*RuleSetName*> to signify that your ruleset depends on another ruleset.

Using the Recursive Feature

Rewriter uses the recursive feature to search to the end of the matched string pattern for the same pattern.

For example, when Rewriter parses the following string:

```
<a href="src=abc.jpg,src=bcd.jpg,src=xyz.jpg>
```

the rule

```
<Attribute name="href" valuePatterns="*src=**"/>
```

rewrites only the first occurrence of the pattern, which would look like this:

```
<a href="src=http://jane.sun.com/abc.jpg>
```

If you use the recursive option

```
<Attribute name="href" valuePatterns="REC:*src=**"/>;
```

Rewriter searches to the end of the matched string pattern for the same pattern, so the output would be:

```
<a
href="src=http://jane.sun.com/abc.jpg,src=http://jane.sun.com/bcd.jpg,src=http://jane
```

Defining Language Based Rules

Rules are based on the following languages:

- HTML
- JavaScript
- XML

Rules for HTML Content

HTML content in web pages can be further classified into attributes, forms and applets. Accordingly, the rules for HTML content are classified as:

- "Attribute Rules for HTML Content" on page 31
- "Form Rules for HTML Content" on page 33
- "Applet Rules for HTML Content" on page 35

Attribute Rules for HTML Content

This rule identifies the attributes of a tag whose value needs to be rewritten. The attribute values can be a simple URL, JavaScript, or DHTML content. For example:

- `src` attributes of an "img" tag point to an image location (simple URL)
- `onClick` attribute of a href attributes that handles on clicking of the link (DJS)

This section describes the following:

- "Attribute Rule Syntax" on page 32
- "Attribute Rule Example" on page 32
- "DJS Attribute Example" on page 33

Attribute Rule Syntax

```
<Attribute name="attributeName" [tag="*" valuePatterns="" source="*" type="URL|DHTML|DJS"]/>
```

where,

`attributeName` is the name of the attribute (mandatory)

`tag` is the tag to which the attribute belongs (optional, default * , meaning any tag)

`valuePatterns` See "Using Pattern Matching in Rules" on page 36.

`source` specifies the URI of the page in which this attribute is defined (optional, default * , meaning in any page)

`type` specifies the type of the value (optional). They can be:

`URL` - a simple URL (default value).

`DHMTL` - DHTML content. This kind of content is seen in standard HTML content and is used in Microsoft's HTC format files.

`DJS` - JavaScript content. All HTML event handlers such as onClick and onMouseover have JavaScript inlined with the HTML attribute.

Attribute Rule Example

Assume the base URL of the page is:

```
http://mymachine.intranet.com/mypage.html
```

Page Content:

```
<a href="http://mymachine.intranet.com/mypage.html">
```

Rules

```
<Attribute name="href"/>
or
<Attribute name="href" tag="a"/>
```

Output

```
<a href=gateway-URL/http://mymachine.intranet.com/myhome.html>
```

Description

Because the URL to be rewritten is already an absolute URL, only the Gateway URL is prefixed to the URL.

DJS Attribute Example

Assume the base URL of the page is:

```
http://abc.sesta.com/focus.html
```

Page Content:

```
<Form>

<input TYPE=TEXT SIZE=20 value=focus
onClick="Check(\q/focus.html\q,\qfocus\q);return;">

</Form>
```

Rules

```
<Attribute name=”onClick” type=”DJS”/>
<Function type="URL" name="Check" paramPatterns="y,"/>
```

Output

```
<Form>

<INPUT TYPE=TEXT SIZE=20 value=focus onClick="Check(\q
gateway-URL
/http://abc.sesta.com/focus.html\q,\qfocus\q);return;">

</Form>
```

Description

Two rules are required to rewrite the specified page content. The first rule identifies the `onClick` JavaScript token. The second rule identifies the parameter of the `check` function that needs to be rewritten. In this case, only the first parameter is rewritten because `paramPatterns` has the value `y` in place of first parameter.

The Gateway URL and the base URL of the page on which the JavaScript tokens appear are prefixed to the required parameter.

Form Rules for HTML Content

The HTML pages that a user browses may contain forms. Some form elements may take a URL as the value.

This section is divided into the following parts:

- “Form Rule Syntax” on page 34
- “Form Rule Example” on page 34

Form Rule Syntax

```
<Form name="form1" field="visit" [valuePatterns="" source="*"]/>
```

where

`name` is the name of the form (mandatory)

`field` is the field in the form whose value needs to be rewritten (mandatory)

`valuePatterns` See "Using Pattern Matching in Rules" on page 36

`source` is the URL of the html page where this form definition is present (optional, default *, meaning in any page)

Form Rule Example

Assume the base URL of the page is:

```
http://test.siroe.com/testcases/html/form.html
```

Page Content

Assume the page URI is `form.html` and is located in the root directory of the server.

```
<form name=form1  method=POST action=
"http://test.siroe.com/testcases/html/form.html">
<input type=hidden name=abc1 value="0|1234|/test.html">
</form>
```

To rewrite `/text.html` present in the value of hidden field named `abc1` which is part of `form1`. The following rules are needed.

Rules

```
<Form source="*/form.html" name="form1"
field="abc1" valuePatterns="0|1234|"/>
<Attribute name="action"/>
```

Output

```
<FORM name=”form1”
method=”POST” action="gateway-URL/
http://test.siroe.com/testcases/html/form.html">
<input type=hidden name=abc1
value="0|1234|gateway-URL/
http://test.siroe.com/test.html">
</FORM>
```

Description

The `action` tag is rewritten using some defined HTML attribute rule.

The input tag attribute value's `value` is rewritten as shown in the output. The specified `valuePatterns` is located, and all content following the matched `valuePatterns` is rewritten by prefixing the Gateway URL, and the base URL of the page. See "Using Pattern Matching in Rules" on page 36.

Applet Rules for HTML Content

A single web page may contain many applets, and each applet may contain many parameters. Rewriter matches the values specified in the rule with the HTML definition of the applet and modifies the URL values present as a part of the applet parameter definition. This replacement is carried out at the server and not when the user is browsing the particular web page. This rule identifies and rewrites the parameters in both the applet and object tags of the HTML content.

This section is divided into the following parts:

- "Applet Rule Syntax" on page 35
- "Applet Rule Example" on page 35

Applet Rule Syntax

```
<Applet code="ApplicationClassName/ObjectID
" param="parametername" [valuePatterns="" source="*"] />
```

where

`code` is the name of the applet or object class (mandatory)

`param` is the name of the parameter whose value needs to be rewritten (mandatory)

`valuePatterns` See "Using Pattern Matching in Rules" on page 36.

`source` is the URL of the page that contains the applet definition (optional, default is *, meaning, in any page)

Applet Rule Example

Assume the base URL of the page is:

```
http://abc.siroe.com/casestudy/test/HTML/applet/rule1.html
```

Page Content:

```
<applet codebase=”appletcode” code=”
RewriteURLinApplet.class” archive=”/test.jar”>
<param name=Test1 value="/index.html">
</applet>
```

Rules

```
<Applet source="*/rule1.html" code=
"RewriteURLin*.class" param="Test*"/>
```

Output

```
<APPLET codebase="gateway-URL
/http://abc.siroe.com/casestudy/test/HTML/
applet/appletcode" code="RewriteURLinApplet.class"
 archive="/test.jar"><param name="Test1" value="
gateway-URL/http:
//abc.siroe.com/index.html">
</APPLET>
```

Description

The `codebase attribute` is rewritten because `<Attribute name="codebase"/>` is a defined rule in the `default_gateway_ruleset`.

All parameters whose names begin with `Test` are rewritten. The base URL of the page on which the applet code displays and the Gateway URL are prefixed to the value attribute of the param tag.

Using Pattern Matching in Rules

You can use the `valuePatterns` field to achieve pattern matching and identify the specific parts of a statement that need to be rewritten.

If you have specified `valuePatterns` as part of a rule, all the content that follows the matched pattern is rewritten.

Consider the sample form rule below.

```
<Form source="*/source.html
" name="form1" field="visit
" [valuePatterns="0|1234|"]/>
```

where

`source` is the URL of the html page where the form displays.

`name` is the name of the form.

`field` is the field in the form whose value needs to be rewritten.

`valuePatterns` indicates the portion of the string that needs to be rewritten. All content appearing after `valuePatterns` is rewritten (optional, default "" means the full value needs to be rewritten).

Specifying Specialized Characters in valuePatterns

You can specify specialized characters by escaping them with a backslash. For example:

```
<Form source="*/source.html" name="form1" field="visit"
[valuePatterns="0|1234|\\;original text|changed text"]/>
```

Using Wild Cards in valuePatterns

You can use the wildcard asterisk (*) character to achieve pattern matching for rewriting.

You cannot specify just an * in the `valuePatterns` field. Because * indicates a match with all text, no text follows the `valuePattern`. Therefore, Rewriter has no text to rewrite. You must use * in conjunction with another string such as `*abc`. In this case, all content that follows `*abc` is rewritten.

Note – An asterisk (*) can be used as a wildcard in any of the fields of the rule. However, all the fields in the rule cannot contain an *. If all fields contain a *, the rule is ignored. No error message is displayed.

You can use a * or ** along with the separation character (a semicolon or comma) that displays in the original statement to separate multiple fields. One asterisk (*) matches any field that is not to be rewritten, and two asterisks (**) to match any field that needs to be rewritten.

"Using Wild Cards in valuePatterns" on page 37 lists some sample usages of the * wildcard.

TABLE 4–1 Sample Usage of * Wildcard

URL	valuePatterns	Description
`url1, url2, url3, url4`	`valuePatterns = "**, *, **, *"`	`url1` and `url3` are rewritten because ** indicates the portion to be rewritten
`XYZABChttp://host1.sesta.com/dir1.html`	`valuePatterns = "*ABC"`	only the portion `http://host1.sesta.com/dir1.html` is rewritten. Everything after `*ABC` needs to be rewritten.
`"0|dir1|dir2|dir3|dir4|test|url1`	`valuePatterns = "*|*|**|*|**|*|"`	`dir2`, `dir4` and `url1` are rewritten. The last field that needs to be rewritten does not have to be indicated by using **.

Rules for JavaScript Content

JavaScript can contain URLs in various locations. Rewriter cannot directly parse the JavaScript and determine the URL portion. A special set of rules need to be written to help the JavaScript processor to identify and translate the URL.

JavaScript elements with type URL are classified as follows:

- “Variables” on page 38
- “Function Arguments” on page 44

Variables

The generic syntax for variables is:

<Variable name=”variableName“ [type="URL|EXPRESSION|DHTML|DJS|SYSTEM" source=”*”]>

JavaScript variables can be subclassified into 5 categories depending on the type of value they hold:

- “URL Variables” on page 38
- “EXPRESSION Variables” on page 39
- “DHTML(Dynamic HTML) Variables” on page 41
- “DJS (Dynamic JavaScript) Variables” on page 42
- “SYSTEM Variables” on page 43

URL Variables

The variable value is a simple string which can be treated as a URL.

This section is divided into the following parts:

- “URL Variable Syntax” on page 38
- “URL Variable Example” on page 39

URL Variable Syntax

```
<Variable name=”variableName“ type="URL" [source=”*”]>
```

where

`variableName` is the name of the variable. The value of the `variableName` is rewritten (mandatory).

`type` is the URL variable (mandatory, and the value must to be a URL)

`source` is the URI of the page in which this JavaScript variable is found (optional, default is *, meaning in any page)

URL Variable Example

Assume the base URL is:

```
http://abc.siroe.com/tmp/page.html
```

Page Content

```
<script LANGUAGE="Javascript">
<!--
//URL Variables
var imgsrc1="/tmp/tmp.jpg";
var imgsrc2="http://srap.sesta.com/tmp/tmp.jpg";
var imgsrc3=imgsrc2;
//-->
</SCRIPT>
```

Rules

```
<Variable name=”imgsrc*” type="URL"/>
```

Output

```
<script LANGUAGE="Javascript">
<!--
//URL Variables
var imgsrc="gateway-URL/http://abc.siroe.com/tmp/tmp.jpg";
var imgsrc="gateway-URL/http://srap.sesta.com/tmp/tmp.jpg";
var imgsrc3=imgsrc2;
//-->
</SCRIPT>
```

Description

All variables of type URL and name beginning with `imgsrc` are rewritten. For the first line of the output, the Gateway URL and the base URL of the page on which the variable displays are prefixed. The second line already contains the absolute path, and hence only the Gateway URL is prefixed. Third var `imagsrc2` would not be rewritten as it's value is not a string but another JavaScript value.

EXPRESSION Variables

Expression variables have an expression on the right hand side. The result of this expression is a URL. Rewriter appends a JavaScript function (`psSRAPRewriter_convert_expression`) to the HTML page as it cannot evaluate such expressions on the server. This function takes the expression as a parameter and evaluates it to the required URL at the client browser.

If you are not sure whether a statement contains a simple URL or an EXPRESSION URL, use EXPRESSION rules because it can handle both scenarios.

This section is divided into the following parts:

EXPRESSION Variable Syntax

```
<Variable name="variableName" [type="EXPRESSION" source="*"]/>
```

where

`variableName` is the name of the JavaScript variable whose value is a expression (mandatory)

`type` is the type of JavaScript variable (optional, default value is EXPRESSION)

`source` is the URI of the pages (optional, default is *, meaning any source)

EXPRESSION Variable Example

Assume the base URL of the page is:

```
http://abc.siroe.com/dir1/dir2/page.html
```

Page Content

```
<script LANGUAGE="Javascript">
<!--
//Expression variables
var expvar= getURIPreFix() + "../../images/graphics"+".gif";
document.write("<A HREF="+expvar+">Link to XYZ content</A><P>")
var expvar="../../images/graphics"+".gif";
//-->
</SCRIPT>
```

Rules

```
<Variable name="expvar" type="EXPRESSION"/>
or
<Variable name="expvar"/>
```

Output

```
var expvar=psSRAPRewriter_convert_expression(getURIPreFix()
 + "../../images/graphics"+".gif");document.write("<a href="+expvar+">>
Link to XYZ content</A><P>")var expvar="gateway-URL/http://abc.siroe.com/images/graphics"+".gif";
```

Description

The function `psSRAPRewriter_convert_expression` is prefixed to the right side of the expression variable `expvar` in the first line. This function processes the expression and rewrites the content at runtime. In the third line the value is rewritten as a simple URL.

DHTML(Dynamic HTML) Variables

These are JavaScript variables that contain HTML content.

This section is divided into the following parts:

- "DHTML Syntax" on page 41
- "DHTML Example" on page 41

DHTML Syntax

```
<Variable name=”variableName” type="DHTML" [source=”*”]/>
```

where

`variableName` is the name of the JavaScript variable with DHTML content (mandatory)

`type` is the type of the variable (mandatory, the value must be DHTML)

`source` is the URL of the page (optional, the default is *, meaning in any page)

DHTML Example

Assume the base URL of the page is:

```
http://abc.sesta.com/graphics/set1/
graphics/jsscript/JSVAR/page.html
```

Page Content

```
<script LANGUAGE="Javascript">
<!--
//DHTML Var
var dhtmlVar="<a href=../../images/test.html>"
var dhtmlVar="<a href=/images/test.html>"
var dhtmlVar="<a href=images/test.html>"
//-->
</SCRIPT>
```

Rules

```
<Variable name=”dhtmlVar” type="DHTML"/>
<Attribute name="href"/>
or
<Attribute name="href" tag="a"/>
```

Output

```
<script LANGUAGE="Javascript">
<!--
//DHTML Var
var dhtmlVar="<a href=gateway-URL
/http://abc.sesta.com/graphics/
set1/graphics/images/test.html>"
var dhtmlVar="<a href=gateway-URL/
http
://abc.sesta.com/images/test.html>"
var dhtmlVar="<a href=gateway-URL/
http://abc.sesta.com/graphics/set1/
graphics/jscript/JSVAR/images/test.html>"
//--></SCRIPT>
```

Description

The JavaScript parser reads the value of `dhtmlVar` as HTML content and sends the content through the HTML parser. The HTML parser applies the HTML rules where the href attribute rules are matched and hence the URL is rewritten.

DJS (Dynamic JavaScript) Variables

These are JavaScript variables that contain JavaScript content.

This section is divided into the following parts:

- "DJS Syntax" on page 42
- "DJS Example" on page 42

DJS Syntax

```
<Variable name="variableName" type="DJS" [source="*"]/>
```

where

`variable` is the JavaScript varible whose value is javascript.

DJS Example

Assume the base URL of the page is:

```
http://abc.sesta.com/dir1/dir2/dir3/jscript/dir4/page.html
```

Page Content

```
//DJS Var
var dJSVar="var dJSimgsrc=\q/tmp/tmp.jpg\q;"
var dJSVar="var dJSimgsrc=\q../tmp/tmp.jpg\q;"
var dJSVar="var dJSimgsrc=
\qhttp://abc.sesta.com/tmp/tmp.jpg\q;"
```

Rules

```
<Variable name="dJSVar" type="DJS"/>
<Variable name="dJSimgsrc" type=URL"/>
```

Output

```
//DJS Var - need 2 rules
var dJSVar="var dJSimgsrc=\qgateway-URL
/http://abc.sesta.com/tmp/tmp.jpg\q;"var dJSVar="var dJSimgsrc=\q
gateway-URL/http
://abc.sesta.com/dir1/dir2/dir3/jscript/tmp/tmp.jpg\q;"
var dJSVar="var dJSimgsrc=\qgateway-URL/
http://abc.sesta.com/tmp/tmp.jpg\q;"
```

Description

Two rules are required here. The first rule locates the dynamic JavaScript variable `dJSVar`. The value of this variable is again a JavaScript of type `URL`. The second rule is applied to rewrite the value of this JavaScript variable.

SYSTEM Variables

These are variables are not declared by the use and have limited support. They are available as a part of the JavaScript standard. For example, `window.location.pathname`.

This section is divided into the following parts:

- "SYSTEM Variable Syntax" on page 43
- "SYSTEM Variable Example" on page 44

SYSTEM Variable Syntax

```
<Variable name="variableName" type="SYSTEM" [source="*"]/>
```

where

`variableName` is the JavaScript system variable (mandatory and the values could be ones that match these patterns: document.URL, document.domain, location, doument.location, location.pathname, location.href, location.protocol, location.hostname, location.host and location.port. All these are present in the generic_ruleset. Do not modifiy these system var rules
.

`type` specifies system type values (mandatory and value is SYSTEM)

`source` is the URI of this pages (optional, default value is *, meaning in any page)

SYSTEM Variable Example

Assume the base URL of the page is:

```
http://abc.siroe.com/dir1/page.html
```

Page Content

```
<script LANGUAGE="Javascript">
<!--
//SYSTEM Var
alert(window.location.pathname);
//-->
</SCRIPT>
```

Rules

```
<Variable name=”window.location.pathname” type="SYSTEM"/>
```

Output

```
</SCRIPT>
<SCRIPT LANGUAGE="Javascript">
<!--
//SYSTEM Var
alert(psSRAPRewriter_convert_system(window.location.pathname));
//-->
</SCRIPT>
```

Description

Rewriter locates the system variable which matches the rule, then the `psSRAPRewriter_convert_system` function is prefixed. This function processes the system variable at runtime and rewrites the resulting URL accordingly.

Function Arguments

Function parameters whose value needs to be rewritten are classified into 4 categories:

- “URL Parameters” on page 45
- “EXPRESSION Parameters” on page 47
- “DHTML Parameters” on page 48
- “DJS Parameters” on page 50

Generic Syntax

```
<Function name="functionName" paramPatterns="y,y,"
[type="URL|EXPRESSION|DHTML|DJS" source="*"]/>
```

where

`name` is the name of the JavaScript function (mandatory)

`paramPatterns` specifies the parameters that need to be rewritten (mandatory)

`y` the position of `y` indicates the parameter that the needs to be rewritten. For example, in the syntax, the first parameter needs to be rewritten, but the second parameter should not be rewritten.

`type` specifies the kind of value this parameter needs (optional, default is EXPRESSION type)

`source` page source URI (optional, default is *, meaning in any page)

URL Parameters

Function takes this parameter as a string and this string could be treated as URL.

This section is divided into the following parts:

- "URL Parameter Syntax" on page 45
- "URL Parameter Example" on page 46

URL Parameter Syntax

```
<Function name="functionName" paramPatterns="y,," type="URL" [source="*"]/>
```

where

`name` is the name of the function with a type parameter of URL (mandatory)

`paramPatterns` specifies the parameters that need to be rewritten (mandatory)

`y` the position of `y` indicates the parameter that needs to be rewritten. For example, in the syntax, the first parameter needs to be rewritten, but the second parameter should not be rewritten.

`type` is the type of the function (mandatory, and the value must be URL)

`source` is the URL of the page which has this function call (optional, default is *, meaning in any URL)

URL Parameter Example

Assume the base URL of the page is:

```
http://abc.sesta.com/test/rewriter/test1/jscript/test2/page.html
```

Page Content

```
<script language="JavaScript">
<!--
function test(one,two,three){
alert(one + "##" + two + "##" +three);
}
test("/test.html","../test.html","123");
window.open("/index.html","gen",width=500,height=500);
//-->
</SCRIPT>
```

Rules

```
<Function type="URL" name="test" paramPatterns="y,y,"/>
<Function type="URL" name="window.open" paramPatterns="y,,,"/>
```

Output

```
<SCRIPT language="JavaScript">
<!--
function test(one,two,three) {
alert(one + "##" + two + "##" +three);
}
test("gateway-URL/http://abc.sesta.com/test.html","
gateway-URL/http://abc.sesta.com/test/rewriter/
test1/jscript/test.html","123");window.open("gateway-URL/
http://abc.sesta.com/index.html","gen",width=500,height=500);
//-->
</SCRIPT>
```

Description

The first rule specifies that the first two parameters in the function with name `test` need to be rewritten. Hence the first two parameters of the test function are rewritten. The second rule specifies that the first parameter of the `window.open` function needs to be written. The URL within the `window.open` function is prefixed with the Gateway URL and the base URL of the page that contains the function parameters.

EXPRESSION Parameters

These parameters take an expression value, which when evaluated, results in a URL.

This section is divided into the following parts:

EXPRESSION Parameter Syntax

```
<Function name="functionName" paramPatterns="y" [type="EXPRESSION" source="*"]/>
```

where

`name` is the name of the function (mandatory).

`paramPatterns` specifies the parameters that need to be rewritten (mandatory)

`y` the position of `y` indicates the function parameter that needs to be rewritten. In the syntax above, only the first parameter is rewritten.

`type` specifies the value EXPRESSION (optional)

`source` URI of the page where this function is called.

EXPRESSION Parameter Example

Assume the base URL of the page is:

```
http://abc.sesta.com/dir1/dir2/page.html
```

Page Content

```
<script language="JavaScript">
<!--
function jstest2(){
return ".html";
}
function jstest1(one){
return one;
}
var dir="/images/test"
var test1=jstest1(dir+"/test"+jstest2());
document.write("<a HREF="+test1+">TEST</a>");
alert(test1);
//-->
</SCRIPT>
```

Rules

```
<Function type="EXPRESSION" name="jstest1" paramPatterns="y"/>
or
<Function name="jstest1" paramPatterns="y"/>
```

Output

```
<script language="JavaScript">
<!--
function jstest2(){
return ".html";
}
function jstest1(one){
return one;
}
var dir="/images/test"
var test1=jstest1(psSRAPRewriter_convert_expression(dir+"/test"+jstest2()));
document.write("<a HREF="+test1+">TEST</a>");
alert(test1);
//-->
</SCRIPT>
```

Description

The rule specifies that the first parameter of the `jstest1` function needs to be rewritten by considering this as an EXPRESSION function param. In the sample page content, the first parameter is an expression that will be evaluated only at runtime. Rewriter prefixes this expression with the `psSRAPRewriter_convert_expression` function. The expression is evaluated, and the `psSRAPRewriter_convert_expression` function rewrites the output at runtime.

Note – In the above example, the variable `test1 is not required` as a part of the JavaScript variable rule. The function rule for `jstest1` takes care of the rewriting.

DHTML Parameters

Function parameter whose value is HTML

Native JavaScript methods such as `document.write()` that generate an HTML page dynamically fall under this category.

This section is divided into the following parts:

- "DHTML Parameter Syntax" on page 49
- "DHTML Parameter Example" on page 49

DHTML Parameter Syntax

```
<Function name="functionName" paramPatterns="y" type="DHTML" [source="*"]/>
```

where

`name` is the name of the function.

`paramPatterns` specifies the parameters that need to be rewritten (mandatory)

`y` the position of `y` indicates the function parameter that needs to be rewritten. In the syntax above, only the first parameter is rewritten.

DHTML Parameter Example

Assume the base URL of the page is:

```
http://xyz.siroe.com/test/rewriter/test1/jscript/JSFUNC/page.html
```

Page Content

```
<script>
<!--
document.write(\q<a href="/index.html">write</a><BR>\q)
document.writeln(\q<a href="index.html">writeln</a><BR>\q)
document.write("http://abc.sesta.com/index.html<BR>")
document.writeln("http://abc.sesta.com/index.html<BR>")
//-->
</SCRIPT>
```

Rules

```
<Function type="DHTML" name="document.write" paramPatterns="y"/>
<Function type="DHTML" name="document.writeln" paramPatterns="y"/>
<Attribute name="href"/>
```

Output

```
<SCRIPT>
<!--
document.write(\q<a href="gateway-URL/
http://xyz.siroe.com/index.html">write</a><BR>\q)
document.writeln(\q<a href="gateway-URL/
http://xyz.siroe.com/test/rewriter/test1/
jscript/JSFUNC/index.html">writeln</a><BR>\q)
document.write("http://abc.sesta.com/index.html<BR>")
document.writeln("http://abc.sesta.com/index.html<BR>")
//-->
</SCRIPT>
```

Description

The first rule specifies that the first parameter in the function `document.write` needs to be rewritten. The second rule specifies that the first parameter in the function `document.writeln` needs to be rewritten. The third rule is a simple HTML rule that specifies that all attributes with the name `href` need to be rewritten. In the example, the DHTML parameter rules identify the parameters in the functions that need to be rewritten. Then the HTML attribute rule is applied to actually rewrite the identified parameter.

DJS Parameters

Function parameters whose value is JavaScript.

This section is divided into the following sections:

- "DJS Parameter Syntax" on page 50
- "DJS Parameter Example" on page 50

DJS Parameter Syntax

```
<Function name="functionName" paramPatterns="y" type="DJS" [source="*"]/>
```

where

`name` is the name of the function where one parameter is DJS (mandatory)

paramPatterns specifies which parameter in the above function is DJS (mandatory)

y the position of y indicates the function parameter that needs to be rewritten. In the syntax above, only the first parameter is rewritten.

`type` is DJS (mandatory)

`source` is the URI of the page (optional, default is *, meaning any URI)

DJS Parameter Example

Assume the base URL of the page is:

```
http://abc.sesta.com/page.html
```

Page Content

```
<script>
menu.addItem(new NavBarMenuItem("All Available Information","JavaScript:top.location=\qhttp://abc.sesta.com\q"));
</script>
```

Rules

```
<Function type="DJS" name="NavBarMenuItem" paramPatterns=",y"/>
<Variable type="URL" name="top.location"/>
```

Output

```
<script>
menu.addItem(new NavBarMenuItem("All Available Information",
"JavaScript:top.location=\qgateway-URL/
http://abc.sesta.com\q"));
</script>
```

Description

The first rule specifies that the second parameter of the function `NavBarMenuItem` which contains JavaScript needs is to be rewritten. Within the JavaScript, the variable `top.location` also needs to be rewritten. This variable is rewritten using the second rule.

Rules for XML Content

Web pages may contain XML content which in turn can contain URLs. XML content that needs to be rewritten is classified into two categories:

- "Tag Text" on page 51 (same as PCDATA or CDATA of the tag)
- "Attribute" on page 52

Tag Text

This rule is for rewriting the PCDATA of CDATA of the tag element.

This section is divided into the following parts:

- "Tag Text Syntax" on page 51
- "Tag Text Example" on page 52

Tag Text Syntax

```
<TagText tag="tagName"
[attributePatterns="attribute_patterns_for_ this_tag" source="*"]/>
```

where

`tagName` is the name of the tag

`attributePatterns` is the attributes and their value patterns for this tag (optional, meaning this tag has no attributes at all)

`source` is the URI of this xml file (optional, default is *, meaning, any xml page)

Tag Text Example

Assume the base URL of the page is:

```
http://abc.sesta.com/test/rewriter/test1/xml/page.html
```

Page Content

```
<xml>
<Attribute name="src">test.html</attribute>
<attribute>abc.html</attribute>
</xml>
```

Rules

```
<TagText tag="attribute" attributePatterns="name=src"/>
```

Output

```
<xml>
<Attribute name="src">gateway-URL/
http://abc.sesta.com/test/rewriter/test1/
xml/test.html</attribute><attribute>abc.html</attribute>
</xml>
```

Description

The first line in the page content has an "Attribute Example" on page 53. The second line in the page content does not contain an attribute with the attribute called name and value of attribute name to be `src`, and hence no rewriting is done. To rewrite this also we need to have `<TagText tag="attribute"/>`

Attribute

The rules for XML attributes are similar to the attribute rules for HTML. The difference is that attribute rules of XML are cases sensitive while HTML attribute rules are not. This is again due to case sensitivity built into XML and not into HTML.

Rewriter translates the attribute value based on the attribute name.

This section is divided into the following parts:

Attribute Syntax

```
<Attribute name="attributeName" [tag="*" type=”URL” valuePatterns="*"
source=”*”]/>
```

where

`attributeName` is the name of the attribute (mandatory)

`tag` is the name of the tag, where this attribute is present (optional, deafult is * , meaning any tag)

`valuePatterns` See “Using Pattern Matching in Rules” on page 36.

`source` is the URI of this XML page (optional, default is *, meaning in any XML page)

Attribute Example

Assume the base URL of the page is:

```
http://abc.sesta.com/test/rewriter/test1/xml/page.html
```

Page Content

```
<xml>
<baseroot href="/root.html"/>
<img href="image.html"/>
<string href="1234|substring.html"/>
<check href="1234|string.html"/>
</xml>
```

Rules

```
<Attribute name="href"tag="check" valuePatterns="1234|"/>
```

Output

```
<xml>
<baseroot href="/root.html"/><img href="image.html"/>
<string href="1234|substring.html"/><check href="1234|
gateway-URL
/http://abc.sesta.com/test/rewriter/test1/xml/string.html"/></xml>
```

Description

In the above example, only the fourth line is rewritten because it meets all the conditions specified in the rule. See “Using Pattern Matching in Rules” on page 36.

Rules for Cascading Style Sheets

The Cascading Style Sheets (including CCS2) in HTML pages are translated. No rules are defined for this translation as the URL presents only in the `url()` functions and import syntaxes of the CSS.

Rules for WML

WML is similar to HTML and hence HTML rules are applied for WML content.Use the generic ruleset for WML content. See "Rules for HTML Content" on page 31.

Using the Recursive Feature

Rewriter uses the recursive feature to search to the end of the matched string pattern for the same pattern.

For example, when Rewriter parses the following string:

```
<a href="src=abc.jpg,src=bcd.jpg,src=xyz.jpg>
```

the rule

```
<Attribute name="href" valuePatterns="*src=**"/>
```

rewrites only the first occurrence of the pattern and it would look like this:

```
<a href="src=http://jane.sun.com/abc.jpg>
```

but if you use the recursive option as,

```
<Attribute name="href" valuePatterns="REC:*src=**"/>;
```

Rewriter searches to the end of the matched string pattern for the same pattern, hence the output would be:

```
<a
href="src=http://jane.sun.com/abc.jpg,src=http://jane.sun.com/bcd.jpg,src=http://jane.sun.
```

Troubleshooting Using Debug Logs

To troubleshoot a Rewriter problem, you need to enable debug logs.

Debug Messages are classified as follows.

- Error– errors that Rewriter cannot recover from
- Warning– warnings that do not critically affect the functioning of Rewriter. Rewriter is able to recover this type of error, but some misbehavior may or may not result. Some messages shown in warnings are informational. For example "Not rewriting image content" is logged as a warning message. This is fine as Rewriter is not supposed to rewrite the images.
- Message– the highest level of information that Rewriter provides.

Setting the Rewriter Debug Level

▼ To Set the Rewriter Debug Level

1 Log in as root to the Gateway machine and edit the following file:

gateway-install-root`/SUNWam/config/AMConfig-`*instance-name*`.properties`

2 Set the debug level:

```
com.iplanet.services.debug.level=
```

The debug levels are:

`error` - Only serious errors are logged in the debug file. Rewriter usually stops functioning when such errors occur.

`warning` - Warning messages are logged.

`message` - All debug messages are logged.

`off` - No debug messages are logged.

3 Specify the directory for the debug files in the following property of the `AMConfig-`*instance-name*`.properties` file:

```
com.iplanet.services.debug.directory=/var/opt/SUNWam/debug
```

where `/var/opt/SUNWam/debug` is the default debug directory.

4 Restart the Gateway from a terminal window:

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

Debug File Names

When the debug level is set to message, debug generates a set of files. "Debug File Names" on page 56 lists the Rewriter files and the information contained within them.

TABLE 4–2 Rewriter Debug Files

File Name	Information
RuleSetInfo	Contains all the rulesets which have been used for rewriting, are logged in this file.
Original Pages	Contains the page URI, resolveURI (if different than the page URI), content MIME, the ruleset that has been applied to the page, parser MIME, and the original content. Specific error/warning/messages related to parsing also appear in this file. In message mode full content is logged. In warning and error mode only exceptions that occurred during rewriting are logged.
Rewritten Pages	Contains the page URI, resolveURI (if different than the page URI), content MIME, ruleset that has been applied to the page, parser MIME, and the rewritten content. This is filled when the debug mode is set to message.
Unaffected Pages	Contains a list the pages that were not modified.
URIInfo Pages	Contains the URLs found and translated. Details of all the pages whose content remain same as original data are logged in this file. Details logged are: Page URI, MIME and Encoding data, rulesetID used for rewriting, and Parser MIME.

In addition to the above files, Rewriter generates a file for debug messages that are not captured in the above files. This file name consists of two parts: the first part is either `pwRewriter` or `psSRARewriter` and the second part is an extension using either `portal` or the *gateway-profile-name*.

The debug files are displayed on the portal or the Gateway. These files are in the directory indicated in the `AMConfig-`*instance-name*`.properties` file.

The Rewriter component generates the following set of files to help in debugging,

*prefix*_RuleSetInfo.*extension*

*prefix*_OrginalPages.*extension*

*prefix*_RewrittenPages.*extension*

*prefix*_UnaffectedPages.*extension*

*prefix*_URIInfo.*extension*

where

prefix is either `psRewriter` for URLScraper usage logs or `psSRAPRewriter` for Gateway usage logs.

extension is either `portal` for URLScraper usage or `gateway-profile-name` for Gateway usage.

For example, if the Rewriter on the Gateway is used to convert pages and the default Gateway profile is used, debug creates these files:

```
psSRAPRewriter_RuleSetInfo.default

psSRAPRewriter_OriginalPages.default

psSRAPRewriter_RewrittenPages.default

psSRAPRewriter_UnaffectedPages.default

psSRAPRewriter_URIInfo.default

psSRAPRewriter.default
```

Working Samples

This section includes:

- Simple HTML pages with content that needs to be rewritten
- Rules required to rewrite the content
- Corresponding rewritten HTML page

These sample pages are available in the *portal-server-URL*`/rewriter` directory. You can browse through the page before the rule is applied, and then view the file with the rewritten output through your Gateway to see how the rule works. In some samples, the rule is already a part of the `default_gateway_ruleset`. In some samples, you may have to include the rule in the `default_gateway_ruleset`. This is mentioned at the appropriate places.

Note – Some of the statements appear in bold to indicate that they have been rewritten.

The following samples are available:

HTML

- "Sample for HTML Attributes" on page 58
- "Sample for HTML Forms" on page 62
- "Sample for HTML Applets" on page 64

JavaScript

- Variables
  - “Sample for JavaScript URL Variables” on page 66
  - “Samples for JavaScript Content” on page 66
  - “Sample for JavaScript DHTML Variables” on page 70
  - “Sample for JavaScript DJS Variables” on page 72
  - “Sample for JavaScript SYSTEM Variables” on page 74

Functions

- “Sample for JavaScript URL Functions” on page 76
- “Sample for JavaScript EXPRESSION Functions” on page 77
- “Sample for JavaScript DHTML Functions” on page 79
- “Sample for JavaScript DJS Functions” on page 81

XML

- Sample for XML Attributes

Samples for HTML Content

Sample for HTML Attributes

▼ To Use the HTML Attributes Sample

1 **This sample can be accessed from:**

portal-server-URL/rewriter/HTML/attrib/attribute.html

2 **Ensure that `abc.sesta.com` and `host1.siroe.com` are defined in the Proxies for Domains and Subdomains list in the Gateway service.**

If this is not defined, a direct connection is assumed, and the Gateway URL is not prefixed.

You need not add the rule specified in this sample to the `default_gateway_ruleset` because the rule is already defined.

HTML Before Rewriting

```
<html>
Rewriting starts
<head>
<title>TEST PAGE () </title>
</head>
ID-htmlattr.1
```

```
<br><br>
1.a href <a href="http://abc.sesta.com/images/logo.gif">http://..</a>
<br><br>
2. href <a href="https://host1.siroe.com">https://..</a>
<br><br>
3. href <a href="../images/logo.gif">../images/</a>
<br><br>
4. href <a href="images/logo.gif">images/..</a> <br><br>
5. href <a href="../../images/logo.gif">../../images/</a> <br><br>
Rewriting ends
</html>
```

Rule

```
<Attribute name="href"/>
```

HTML After Rewriting

```
<html>
Rewriting starts
<head>
<title>TEST PAGE () </title>
</head>
ID-htmlattr.1
<br><br>
1. a href <a href="gateway-URL/http://abc.sesta.com/images/logo.gif">http://..</a> <br>
```

// This URL is rewritten because the `<Attrib name="href"/>` rule is already defined in the `default_gateway_ruleset`. Because the URL is already absolute, only the Gateway URL is prefixed. Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list in the Gateway service. Otherwise, the Gateway URL is not prefixed, because a direct connection is assumed.

```
2. href <a href="gateway-URL/https://host1.siroe.com">https://..</a>
```

// Again, `host1.siroe.com` needs to be defined in the Proxies for Domains and Subdomains list in the Gateway service. Otherwise, the Gateway URL is not prefixed, because a direct connection is assumed.

```
<br><br>
3. href <a href="gateway-URL/portal-server-URL/rewriter/HTML/images/logo.gif">../images/</a>
```

// Because a relative path is specified, the Gateway URL and the portal-server-URL are prefixed along with the required subdirectories. This link will not work because a directory called `images` under the `HTML` directory is not specified in the sample structure provided.

```
<br><br>
```

```
4 href <a href="gateway-URL/portal-server-URL/rewriter/HTML/attrib/images/
logo.gif">images/..</a> <br><br>
```

// Because a relative path is specified, the Gateway URL and the Portal Server URL are prefixed along with the required subdirectories.

```
5. href <a href="gateway-URL/portal-server-URL/rewriter/images/logo.gif">
../../images/</a> <br><br>
```

// Because a relative path is specified, the Gateway URL and the Portal Server URL are prefixed along with the required subdirectories. This link will not work because a directory called `images` under the `Rewriter` directory is not specified in the sample structure provided

```
Rewriting ends</html>
```

Sample for HTML Dynamic JavaScript Tokens

This section discuses using the HTML JavaScript token sample

▼ To Use the HTML JavaScript Token Sample:

1. **This sample can be accessed from:**
   portal-server-URL/rewriter/HTML/jstokens/JStokens.html
2. **Add the rule specified in this sample to the** `default_gateway_ruleset` **in the section "Rules for Rewriting JavaScript Source".**
3. **Edit the** `default_gateway_ruleset` **in the Rewriter service under the Portal Server Configuration in the Portal Server administration console.**
4. **Restart the Gateway from a terminal window:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Before Rewriting

```
<html>
<head>
Rewriting starts
<script language="javascript">
function Check(test,ind){
if (ind == \qblur\q)
{alert("testing onBlur")}
if (ind == \qfocus\q)
{alert("testing onFocus")}
}
```

```
</SCRIPT>
</head>
<body>
<form>
<input TYPE=TEXT SIZE=20 value=blur onAbort="Check
(\q/indexblur.html\q,\qblur\q);return;">
<input TYPE=TEXT SIZE=20 value=blur onBlur="Check
(\q/indexblur.html\q,\qblur\q);return;">
<input TYPE=TEXT SIZE=20 value=focus onFocus="Check
(\q/focus.html\q,\qfocus\q);return;">
<input TYPE=TEXT SIZE=20 value=focus onChange="Check
(\q/focus.html\q,\qfocus\q);return;">
<input TYPE=TEXT SIZE=20 value=focus onClick="Check
(\q/focus.html\q,\qblur\q);return;">
<br><br>
</form>
</body>
Rewriting ends
</html>
```

Rule

```
<Attribute name=”onClick” type=”DJS”/>
<Function type="URL" name="Check" paramPatterns="y"/>
```

Note – `<Function type="URL" name="Check" paramPatterns="y"/>` is a JavaScript function rule and is explained in detail in the JavaScript function sample.

HTML After Rewriting

```
<html>
<head>
Rewriting starts
<script language="javascript">
function Check(test,ind){
if (ind == \qblur\q)
{alert("testing onBlur")}
if (ind == \qfocus\q)
{alert("testing onFocus")}
}
</SCRIPT>
</head>
<body>
<form>
<input TYPE=TEXT SIZE=20 value=blur onAbort="Check
(\qgateway URL/portal-server-URL/indexblur.html\q,\qblur\q);return;">
<input TYPE=TEXT SIZE=20 value=blur onBlur="Check
```

```
(\qgateway URL/portal-server-URL/indexblur.html\q,\qblur\q);return;">
<input TYPE=TEXT SIZE=20 value=focus onFocus="Check
(\qgateway URL/portal-server-URL/focus.html\q,\qfocus\q);return;">
<input TYPE=TEXT SIZE=20 value=focus onChange="Check
(\qgateway URL/portal-server-URL/focus.html\q,\qfocus\q);return;">
<input TYPE=TEXT SIZE=20 value=focus onClick="Check
(\qgateway URL/portal-server-URL/focus.html\q,\qblur\q);return;">
```

// All the statements are rewritten in this sample. The Gateway and Portal Server URLs are prefixed in each case. This is because rules for `onAbort`, `onBlur`, `onFocus`, `onChange`, and `onClick` are defined in the `default_gateway_ruleset` file. Rewriter detects the JavaScript tokens and passes it to the JavaScript function rules for further processing. The second rule listed in the sample tells Rewriter which parameter to rewrite.

```
</body>
<br>

Rewriting ends

</html>
```

Sample for HTML Forms

▼ To Use the Form Sample

1. **Access the sample from:**

   portal-server-URL/rewriter/HTML/forms/formrule.html

2. **Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list in the Gateway service.**

   If this is not defined, a direct connection is assumed, and the Gateway URL is not prefixed.

3. **Add the rule specified in this sample to the `default_gateway_ruleset` in the section "Rules for Rewriting HTML Attributes".**

4. **Edit the `default_gateway_ruleset` in the Rewriter service under the Portal Server Configuration in the Portal Server administration console.**

5. **Restart the Gateway from a terminal window:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
</head>
<body>
RW_START
<p>
<form name="form1" method="Post" action=
"http://abc.sesta.com/casestudy/html/form.html">
<input type="hidden" name="name1" value="0|1234|/test.html">
<input type="hidden" name="name3" value="../../html/test.html">
<form name="form2" method="Post" action="
http://abc.sesta.com/testcases/html/form.html"><br>
<input type="hidden" name="name1" value="0|1234|
../../html/test.html"></form>
RW_END </p>
</body>
</html>
```

Rule

```
<Form source="*" name="form1" field="name1" valuePatterns="0|1234|"/>
```

HTML Page After Rewriting

```
<HTML>
<HEAD>
RW_START
</HEAD>
<BODY>
<P>
<FORM name=form1  method=POST
action="gateway-URL/http://abc.sesta.com/casestudy/html/form.html">
```

// This URL is rewritten because `<Attribute name="action"/>` is defined as part of the HTML rules in the `default_gateway_ruleset`. Because the URL is already absolute, only the Gateway URL needs to be prefixed. Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list in the Gateway service. Else, the Gateway URL is not prefixed because a direct connection is assumed.

```
<input type=hidden name=name1 value=
"0|1234|gateway URL/portal-server-URL/test.html">
```

// Here the form name is `form1`, and the field name is `name1`. This matches the form name and field name specified in the rule. The rule states the `valuePatterns` as `0|1234|` which matches

the value in this statement. Hence the URL occurring after the valuePattern is rewritten. The Portal Server URL and the Gateway URL are prefixed. See ""Using Pattern Matching in Rules" on page 36 for details on valuePatterns.

```
<input type=hidden name=name3 value="../../html/test.html">
```

// This URL is not rewritten because the name does not match the field name specified in the rule.

```
</FORM>
<FORM name=form2 method=POST action=
"gateway-URL/http://abc.sesta.com/casestudy/html/form.html"><BR>
```

// This URL is rewritten because <Attribute name="action"/> is defined as part of the HTML rules in the default ruleset. Because the URL is already absolute, only the Gateway URL needs to be prefixed.

```
<input type=hidden name=name1 value="0|1234|../../html/test.html">
```

// This URL is not rewritten because the form name does not match the name specified in the rule.

```
</FORM>
</BODY>
RW_END
</HTML>
```

Sample for HTML Applets

▼ To Use the Sample for Applets

1. **Obtain the applet class file. The `RewriteURLinApplet.class` file is present in the following location:**

   portal-server-URL/rewriter/HTML/applet/appletcode

   The base URL of the page where the applet code is present is:

   portal-server-URL/rewriter/HTML/applet/rule1.html

2. **Add the rule specified in this sample to the `default_gateway_ruleset` in the section "Rules for Rewriting HTML Attributes".**

3. **Edit the `default_gateway_ruleset` in the Rewriter service under the Portal Server Configuration in the Portal Server administration console.**

4 **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Before Rewriting

```
<html>
Rewriting starts
<br>
<applet codebase=appletcode code=RewriteURLinApplet.class archive=/test>
<param name=Test1 value="/index.html">
<param name=Test2 value="../index.html">
<param name=Test3 value="../../index.html">
</applet>
Rewriting ends
</html>
```

Rule

```
<Applet source="*/rule1.html" code="RewriteURLinApplet.class" param="Test*" />
```

HTML After Rewriting

```
<HTML>
Rewriting starts
<BR>
<APPLET codebase=gateway-URL/portal-server-URL
/rewriter/HTML/applet/appletcode=RewriteURLinApplet.class archive=/test>
```

// This URL is rewritten because the rule `<Applet name="codebase"/>` is already present as part of the `default_gateway_ruleset` file. the Gateway and the Portal Server URLs are prefixed along with the path up to the `appletcode` directory.

```
<param name=Test1 value=
"gateway-URL/portal-server-URL/index.html">
```

// This URL is rewritten because the base URL of the page is `rule1.html`, and the param name matches the param `Test*` specified in the rule. Because `index.html` is specified to be at the root level, the Gateway and Portal Server URLs are prefixed directly.

```
<param name=Test2 value="gateway-URL
/portal-server-URL/rewriter/HTML/index.html">
```

// This URL is rewritten because the base URL of the page is `rule1.html`, and the param name matches the param `Test*` specified in the rule. The path is prefixed as required.

```
<param name=Test3 value="gateway-URL
/portal-server-URL/rewriter/index.html">
```

// This URL is rewritten because the base URL of the page is `rule1.html`, and the param name matches the param `Test*` specified in the rule. The path is prefixed as required.

```
</APPLET>
Rewriting ends
</HTML>
```

Samples for JavaScript Content

Sample for JavaScript URL Variables

▼ To Use the JavaScript URL Variables Sample

1. **This sample can be accessed from:**
   portal-server-URL/rewriter/JavaScript/variables/url/js_urls.html
2. **Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list in the Gateway service.**
   If this is not defined, a direct connection is assumed, and the Gateway URL is not prefixed.
3. **Add the rule specified in this sample to the `default_gateway_ruleset` in the section "Rules for Rewriting JavaScript Source".**
4. **Edit the `default_gateway_ruleset` in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**
5. **If you added the rule, restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
Rewriting starts
<head>
<title>JavaScript Variable test page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
//URL Variables
var imgsrc="/tmp/tmp.jpg";
var imgsrc="./tmp/tmp.jpg";
```

```
var imgsrc="../tmp/tmp.jpg";
var imgsrc="../../tmp/tmp.jpg";
var imgsrc="http://abc.sesta.com/tmp/tmp.jpg";
var imgsrc="../../../tmp/tmp.jpg";
var imgsrc="tmp/tmp.jpg";
//-->
</SCRIPT>
<br>
Testing JavaScript variables!
<br>
<img src="images/logo.gif">
<br>
Image
</body>
<br>
Rewriting ends
</html>
```

Rule

```
<Variable name=”imgsrc” type="URL"/>
```

HTML Page After Rewriting

```
<html>
Rewriting starts
<head>
<title>JavaScript Variable test page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
//URL Variables
var imgsrc="gateway-URL/portal-server-URL/tmp/tmp.jpg";
var imgsrc="gateway-URL/portal-server-URL
/rewriter/JavaScript/variables/url/tmp/tmp.jpg";
var imgsrc="gateway-URL/portal-server-URL
/rewriter/JavaScript/variables/tmp/tmp.jpg";
var imgsrc="gateway-URL/portal-server-URL
/rewriter/JavaScript/tmp/tmp.jpg";
var imgsrc="gateway-URL/http://abc.sesta.com/tmp/tmp.jpg";
var imgsrc="gateway-URL/portal-server-URL/rewriter/tmp/tmp.jpg";
var imgsrc="gateway-URL/portal-server-URL
/rewriter/JavaScript/variables/url/tmp/tmp.jpg";
```

// All the above URLs are JavaScript variables of type URL and name `imgsrc` as specified in the rule. Hence they are prefixed with the Gateway and the Portal Server URLs. The path following the Portal Server URL is prefixed as required.

```
//-->
</SCRIPT>
<br>
Testing JavaScript variables!
<br>
<img src="gateway URL/portal-server-URL/rewriter
/JavaScript/variables/url/images/logo.gif">
```

// This line is rewritten because the rule `<Attribute name="src"/>` is defined in the `default_gateway_ruleset`

```
<br>
Image
</body>
<br>
Rewriting ends
</html>
```

Sample for JavaScript EXPRESSION Variables

▼ To Use the JavaScript Expression Variables Sample

1. **This sample can be accessed from:**
   portal-server-URL/rewriter/JavaScript/variables/expr/expr.html
2. **Add the rule specified in this sample (if it does not already exist) to the `default_gateway_ruleset` in the section "Rules for Rewriting JavaScript Source".**
3. **Edit the `default_gateway_ruleset` in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**
4. **If you added the rule, restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
<head>
<title>JavaScript EXPRESSION Variables Test Page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
//Expression variables
var expvar1="images";
```

```
var expvar2="/logo.gif";
var expvar = expvar1 + expvar2;
document.write("<A HREF="+expvar+">EXPRESSION</A><P>")
var expvar="/images/logo"+".gif";
document.write("<A HREF="+expvar+">EXPRESSION</A><P>")
//-->
</SCRIPT>
Testing JavaScript EXPRESSION variables
</body>
</html>
```

Rule

```
<Variable type=”EXPRESSION” name="expvar"/>
```

HTML Page After Rewriting

```
<html>
<head>
<title>JavaScript EXPRESSION Variables Test Page</title>
</head>
<body>
<SCRIPT>
// Rewriter appends the wrapper function
psSRAPRewriter_convert_expression here
</SCRIPT>
<script LANGUAGE="Javascript">
<!--
//Expression variables
var expvar1="images";
var expvar2="/logo.gif";
var expvar =psSRAPRewriter_convert_expression( expvar1 + expvar2);
```

// Rewriter recognizes the right hand side of this statement to be a JavaScript EXPRESSION variable. Rewriter is not able to resolve the value of this expression at the server end. Hence, the `psSRAPRewriter_convert_expression` function is prefixed to the expression. The expression is evaluated at the client end, and rewritten as required.

```
document.write("<A HREF="+expvar+">EXPRESSION</A><P>")
```

// The rewritten value of `expvar` from the previous statement is used to arrive at the value of this expression. Because the result is a valid URL (a graphic exists at this location in the sample), the link will work.

```
var expvar="gateway URL/portal-server-URL/images/logo"+".gif";
```

// Rewriter recognizes the right hand side of `expvar` to be a string expression. This can be resolved at the server side, and hence is rewritten directly.

```
document.write("<A HREF="+expvar+">EXPRESSION</A><P>")
```

// The rewritten value of `expvar` from the previous statement is used to arrive at the value of this expression. Because the result is a not a valid URL (a graphic does not exist at the resultant location), the link will not work.

```
//-->
</SCRIPT>
Testing JavaScript EXPRESSION variables
</body>
</html>
```

Sample for JavaScript DHTML Variables

▼ To Use the JavaScript DHTML Variables Sample

1. **This sample can be accessed from:**

   portal-server-URL/rewriter/JavaScript/variables/dhtml/dhtml.html

2. **Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list in the Gateway service. If this is not defined, a direct connection is assumed, and the Gateway URL is not prefixed.**

3. **Add the rule specified in this sample (if it does not already exist) to the `default_gateway_ruleset` in the section "Rules for Rewriting JavaScript Source". Edit the `default_gateway_ruleset` in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**

4. **If you added the rule, restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
<head>
<title>JavaScript DHTML Variable Test Page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
//DHTML Var
var dhtmlVar="<a href=../../images/test.html>"
var dhtmlVar="<a href=/../images/test.html>"
var dhtmlVar="<a href=/images/test.html>"
```

```
var dhtmlVar="<a href=images/test.html>"
var dhtmlVar="<a href=http://abc.sesta.com/images/test.html>"
var dhtmlVar="<img src=http://abc.sesta.com/images/test.html>"
//-->
</SCRIPT>
<br><br>
Testing DHTML Variables
<br><br>
<img src="images/logo.gif">IMAGE
</body>
</html>
```

Rule

```
<Variable name="DHTML">dhtmlVar</Variable>
```

HTML Page After Rewriting

```
<html>
<head>
<title>JavaScript DHTML Variable Test Page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
//DHTML Var
var dhtmlVar="<a href=gateway-URL/portal-server-URL
/rewriter/JavaScript/images/test.html>"
```

// The JavaScript DHTML rule identifies the right hand side of the `dhtmlVar` as dynamic HTML content. Hence, the HTML rules in the `default_gateway_ruleset` file are applied. The dynamic HTML contains a `href` attribute. The `default_gateway_ruleset` defines the rule `<Attribute name="href"/>`. Hence the value of the `href` attribute is rewritten. But the URL is not absolute; therefore, the relative URL is replaced with the base URL of the page, and the required subdirectories. This in turn is prefixed with the Gateway URL to derive the final rewritten output.

```
var dhtmlVar="<a href=gateway-URL
/portal-server-URL/../images/test.html>"
```

// Although the base URL of the page is appended, and the Gateway URL is prefixed, the resultant URL will not work. This is because the initial URL `/../images/test.html` is inaccurate.

```
var dhtmlVar="<a href=gateway-URL
/portal-server-URL/images/test.html>"
```

// Here again, the JavaScript DHTML rule identifies the right hand side to be dynamic HTML content, and passes it to the HTML rules. The HTML rule `<Attribute name="href"/>` from the `default_gateway_ruleset` is applied, and the statement is rewritten as shown. The Gateway URL and Portal Server URL are prefixed.

```
var dhtmlVar="<a href=gateway URL/portal-server-URL/
rewriter/JavaScript/variables/dhtml/images/test.html>"
var dhtmlVar="<a href=gateway URL/http://abc.sesta.com/images/test.html>"
var dhtmlVar="<img src=gateway-URL/
http://abc.sesta.com/images/test.html>"
```

// The JavaScript DHTML rule identifies the dynamic HTML content on the right hand side, and passes the statement to the HTML rules. The `<Attribute name="src"/>` rule in the `default_gateway_ruleset` is applied. Because the URL is absolute, only the Gateway URL needs to be prefixed. Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list for this URL to be rewritten.

```
//-->
</SCRIPT>
<br><br>
Testing DHTML Variables
<br><br>
<img src="gateway-URL/portal-server-URL/
rewriter/JavaScript/variables/dhtml/images/logo.gif">
```

// This line is rewritten because the rule `<Attribute name="src"/>` is defined in the `default_gateway_ruleset`.

```
<br><br>
Image
</body>
</html>
```

Sample for JavaScript DJS Variables

▼ To Use the JavaScript DJS Variables Sample

1. **This sample can be accessed from:**
   portal-server-URL/rewriter/JavaScript/variables/djs/djs.html

2. **Ensure that `abc.sesta.com` is defined in the Proxies for Domains and Subdomains list in the Gateway service. If this is not defined, a direct connection is assumed, and the Gateway URL is not prefixed.**

3 **Add the two rules specified in this sample (if it does not already exist) to the** `default_gateway_ruleset` **in the section "Rules for Rewriting JavaScript Source". Edit the** `default_gateway_ruleset` **in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**

4 **Restart the Gateway:**

`./psadmin start-sra-instance –u amadmin – f` *<password file>* `–N` *<profile name>*`– t` *<gateway>*

HTML Page Before Rewriting

```
<html>
<head>
<title>Dynamic JavaScript Variable Test Page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
var dJSVar="var dJSimgsrc=\q/tmp/tmp/jpg\q;"
var dJSVar="var dJSimgsrc=\q../../../tmp/tmp/jpg\q;"
var dJSVar="var dJSimgsrc=\qhttp://abc.sesta.com/tmp/tmp/jpg\q;"
//-->
</SCRIPT>
<br>
Testing Dynamic JavaScript Variables
<br>
<img src="images/logo.gif">
<br>
Image
</body>
</html>
```

Rule

```
<Variable name=”dJSVar” type="DJS"/>
<Variable name="dJSimgsrc“ type=URL"/>
```

HTML Page After Rewriting

```
<html>
<head>
<title>Dynamic JavaScript Variable Test Page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
var dJSVar="var dJSimgsrc=\qgateway-URL
/portal-server-URL/tmp/tmp/jpg\q;"
```

```
var dJSVar="var dJSimgsrc=\qgateway-URL
/portal-server-URL/rewriter/tmp/tmp/jpg\q;"
var dJSVar="var dJSimgsrc=\qgateway-URL
/http://abc.sesta.com/tmp/tmp/jpg\q;"
```

// All the above statements are rewritten with the Gateway and Portal Server URLs. The required path is prefixed as appropriate. The first rule identifies the right hand side of `dJSVar` as a dynamic JavaScript variable. This is then passed to the second rule which identifies the right hand side of `dJSimgsrc` as a JavaScript variable of type URL. This is rewritten accordingly.

```
//-->
</SCRIPT>
<br>
Testing Dynamic JavaScript Variables
<br>
<img src="gateway-URL/portal-server-URL
/rewriter/JavaScript/variables/djs/images/logo.gif">
```

// This line is rewritten because the rule `<Attribute name="src"/>` is defined in the `default_gateway_ruleset`.

```
<br>
Image
</body>
</html>
```

Sample for JavaScript SYSTEM Variables

▼ To Use the JavaScript System Variables Sample

1. **This sample can be accessed from:**
   portal-server-URL/rewriter/JavaScript/variables/system/system.html
2. **Add the rule specified in this sample (if it does not already exist) to the `default_gateway_ruleset` in the section "Rules for Rewriting JavaScript Source".**
3. **Edit the `default_gateway_ruleset` in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**
4. **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
<head>
<title>JavaScript SYSTEM Variables Test Page</title>
</head>
<body>
<script LANGUAGE="Javascript">
<!--
//SYSTEM Var
alert(window.location.pathname);
//document.write
("<A HREF="+window.location.pathname+">SYSTEM</A><P>")
//-->
</SCRIPT>
Testing JavaScript SYSTEM Variables
<br>
This page displays the path where
the current page is located when loaded.
</body>
</html>
```

Rule

```
<Variable name=”window.location.pathname” type="SYSTEM"/>
```

HTML After Rewriting

```
<html>
<head>
<title>JavaScript SYSTEM Variables Test Page</title>
</head>
<body>
<SCRIPT>
convertsystem function definition...
</SCRIPT>
<script LANGUAGE="Javascript">
<!--
//SYSTEM Var
alert(psSRAPRewriter_convert_system
(window.location, window.location.pathname,”window.location”));
```

// Rewriter identifies `window.location.pathname` as a JavaScript SYSTEM variable. The value of this variable cannot be determined at the server end. So the Rewriter prefixes the variable with the `psSRAPRewriter_convert_pathname` function. This wrapper function determines the value of the variable at the client end and rewrites as required.

```
//-->
</SCRIPT>
Testing JavaScript SYSTEM Variables
<br>
This page displays the path where
the current page is located when loaded.
</body>
</html>
```

Sample for JavaScript URL Functions

▼ To Use the JavaScript URL Functions Sample

1. **This sample can be accessed from:**

   portal-server-URL/rewriter/JavaScript/functions/url/url.html

2. **Add the rule specified in this sample (if it does not already exist) to the** `default_gateway_ruleset` **in the section "Rules for Rewriting JavaScript Source". Edit the** `default_gateway_ruleset` **in the Rewriter service under the Portal Server Configuration in the Portal Server administration console.**

3. **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
<body>
JavaScript URL Function Test Page
<br>
<script language="JavaScript">
<!--
function test(one,two,three)
{
alert(one + "##" + two + "##" +three);
}
test("/test.html","../test.html","123");
window.open("/index.html","gen",width=500,height=500);
//-->
</SCRIPT>
</body>
</html>
```

Rule

```
<Function type="URL" name="test" paramPatterns="y,y"/>
<Function type="URL" name="window.open" paramPatterns="y"/>
```

HTML Page After Rewriting

```
<html>
<body>
JavaScript URL Function Test Page
<br>
<script language="JavaScript">
<!--
function test(one,two,three)
{
alert(one + "##" + two + "##" +three);
}
test("/test.html","../test.html","123");
window.open("gateway-URL/portal-server-URL
/index.html","gen",width=500,height=500);
//-->
</SCRIPT>
</body>
</html>
```

Sample for JavaScript EXPRESSION Functions

▼ To Use the JavaScript Expressions Function Sample

1. **This sample can be accessed from:**

   `<portal-install-location>/SUNWportal/samples/rewriter`

2. **Add the rule specified in this sample (if it does not already exist) to the** `default_gateway_ruleset` **in the section** `Rules for Rewriting JavaScript Source`**.**

3. **Edit the** `default_gateway_ruleset` **in the Rewriter service using the Portal Server administration console.**

4. **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
<body>
JavaScript EXPRESSION Function Test Page
```

```
<br><br><br>
<script language="JavaScript">
<!--
function jstest2()
{
return ".html";
}
function jstest1(one)
{
return one;
}
var dir="/images/test"
var test1=jstest1(dir+"/test"+jstest2());
document.write("<a HREF="+test1+">Test</a>");
alert(test1);
//-->
</SCRIPT>
</body>
</html>
```

Rule

```
<Function type="EXPRESSION" name="jstest1" paramPatterns="y"/>
```

HTML Page After Rewriting

```
<html>
<body>
JavaScript EXPRESSION Function Test Page
<br><br><br>
<script>
<!--
// various functions including psSRAPRewriter_
convert_expression appear here.//-->
</SCRIPT>
<script language="JavaScript">
<!--
function jstest2()
{
return ".html";
}
function jstest1(one)
{
return one;
}
var dir="/images/test"
var test1=jstest1(psSRAPRewriter_convert_
expression(dir+"/test"+jstest2()));
```

// The rule states that the first parameter in the function `jstest1` which is of type EXPRESSION needs to be rewritten. The value of this expression is `/test/images/test.html`. This is prefixed with the Portal Server and the Gateway URLs.

```
document.write("<a HREF="+test1+">Test</a>");
alert(test1);
//-->
</SCRIPT>
</body>
</html>
```

Sample for JavaScript DHTML Functions

▼ To Use the JavaScript DHTML Functions Sample

1. **This sample can be accessed from:**

   portal-server-URL/rewriter/JavaScript/functions/dhtml/dhtml.html

2. **Add the rule specified in this sample (if it does not already exist) to the** `default_gateway_ruleset` **in the section "Rules for Rewriting JavaScript Source".**

3. **Edit the** `default_gateway_ruleset` **in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**

4. **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
<head>
Testing JavaScript DHTML Functions
<br>
<br>
<script>
<!--
document.write(\q<a href="/index.html">write</a><BR>\q)
document.writeln(\q<a href="index.html">writeln</a><BR>\q)
document.write("http://abc.sesta.com/index.html<BR>")
document.writeln("http://abc.sesta.com/index.html<BR>")
//-->
</SCRIPT>
</head>
<body BGCOLOR=white>
<br><br>
```

```
Testing document.write and document.writeln
</body>
</html>
```

Rule

```
<Function type="DHTML" name=" document.write" paramPatterns="y"/>
<Function type="DHTML" name=" document.writeln" paramPatterns="y"/>
```

HTML Page After Rewriting

```
<html>
<head>
Testing JavaScript DHTML Functions
<br>
<br>
<script>
<!--
document.write(\q<a href="gateway-URL
/portal-server-URL/index.html">write</a><BR>\q)
```

// The first rule specifies that the first parameter of the DHTML JavaScript function `document.write` needs to be rewritten. Rewriter identifies the first parameter to be a simple HTML statement. The HTML rules section in the `default_gateway_ruleset` has the rule `<Attribute name="href" />` which indicates that the statement needs to be rewritten.

```
document.writeln(\q<a href="gateway-URL
/portal-server-URL/rewriter/JavaScript/functions/dhtml/index.html">writeln</a><BR>\q)
```

// The second rule specifies that the first parameter of the DHTML JavaScript function `document.writeln` needs to be rewritten. Rewriter identifies the first parameter to be a simple HTML statement. The HTML rules section in the `default_gateway_ruleset` has the rule `<Attribute name="href" />` which indicates that the statement needs to be rewritten.

```
document.write("http://abc.sesta.com/index.html<BR>")
document.writeln("http://abc.sesta.com/index.html<BR>")
```

// The above statements are not rewritten although the DHTML rule identifies the functions `document.write` and `document.writeln`. This is because the first parameter in this case is not simple HTML. It could be any string, and Rewriter does not know how to rewrite this.

```
//-->
</SCRIPT>
</head>
<body BGCOLOR=white>
<br><br>
Testing document.write and document.writeln
```

```
</body>
</html>
```

Sample for JavaScript DJS Functions

▼ To Use the JavaScript DJS Functions Sample

1. **This sample can be accessed from:**
   portal-server-URL/rewriter/JavaScript/functions/djs/djs.html

2. **Ensure that** `abc.sesta.com` **is defined in the Proxies for Domains and Subdomains list in the Gateway service.**
   If this is not defined, a direct connection is assumed, and the Gateway URL is not prefixed.

3. **Add the rule specified in this sample (if it does not already exist) to the** `default_gateway_ruleset` **in the section "Rules for Rewriting JavaScript Source". Edit the** `default_gateway_ruleset` **in the Rewriter service under Portal Server Configuration in the Portal Server administration console.**

4. **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

HTML Page Before Rewriting

```
<html>
Test for JavaScript DJS Functions
<br>
<script>
menu.addItem(new NavBarMenuItem("All Available
Information","JavaScript:top.location=\qhttp://abc.sesta.com\q"));
//menu.addItem(new NavBarMenuItem("All Available Information","http://abc.sesta.com"));
</script>
</html>
```

Rule

```
<Function type="DJS" name="NavBarMenuItem" paramPatterns=",y"/>
<Variable type="URL" name=”top.location”/>
```

HTML Page After Rewriting

```
<html>
Testing JavaScript DJS Functions
<br>
```

```
<script>
menu.addItem(new NavBarMenuItem
("All Available Information","javaScript:top.location=
\qgateway-URL/http://abc.sesta.com\q"));
```

// `abc.sesta.com` is an entry in the Proxies for Domains and Subdomains list in the Gateway service. Hence Rewriter needs to rewrite this URL. But because an absolute URL, the Portal Server URL need not be prefixed. The DJS rule states that the second parameter of the DJS function `NavBarMenuItem` needs to be rewritten. But the second parameter is again a JavaScript variable. A second rule is required to rewrite the value of this variable. The second rule specifies that the value of the JavaScript variable `top.location` needs to be rewritten. Because all these conditions are met, the URL is rewritten.

```
//menu.addItem(new NavBarMenuItem("All Available Information","http://abc.sesta.com"));
```

// Although the DJS rule specifies that the second parameter of the function `NavBarMenuItem` needs to be rewritten, it does not happen in this statement. This is because Rewriter does not recognize the second parameter as simple HTML.

```
</script>
</html>
```

Sample for XML Attributes

▼ To Use the XML Attributes Sample

1. **This sample can be accessed from:**

   portal-server-URL/rewriter/XML/attrib.html

2. **Add the rule specified in this sample (if it does not already exist) to the `default_gateway_ruleset` in the section "Rules for Rewriting XML Source".**

3. **Edit the `default_gateway_ruleset` in the Rewriter service under the Portal Server Configuration in the Portal Server administration console.**

4. **Restart the Gateway:**

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

XML Before Rewriting

```
<html>
RW_START
<body>
```

```
<xml>
<baseroot href="/root.html"/>
</xml>
<xml>
<img href="image.html"/>
</xml>
<xml>
<string href="1234|substring.html"/>
</xml>
<xml>
<check href="1234|string.html"/>
</xml>
</body>
RW_END
</html>
```

Rule

```
<Attribute name="href" tag="check" valuePatterns="1234|"/>
```

HTML After Rewriting

```
<html>
Rewriting starts
<br>
<br>
<body>
<xml><baseroot href="/root.html"/></xml>
<xml><img href="image.html"/></xml>
<xml><string href="1234|substring.html"/></xml>
<xml><check href="1234|gateway-URL/portal-server-URL
/rewriter/XML/string.html"/></xml>
```

// This statement is rewritten because it matches the conditions specified in the rule. The `Attribute name` is `href`, `tag` is `check` and the `valuePatterns` is `1234`. The string following `valuePatterns` is rewritten. See "Using Pattern Matching in Rules" on page 36 for details on `valuePatterns`.

```
</body>
Rewriting ends
</html>
```

Case Study

This section includes the source HTML pages for a sample mail client. This case study does not cover all possible scenarios and rules. This is just a sample ruleset to help you put together the rules for your intranet pages.

Assumptions

The following assumptions are made for this case study:

- The base URL of the mail client is assumed to be `abc.siroe.com`
- The Gateway URL is assumed to be `gateway.sesta.com`
- Relevant entries exist in the Proxies for Domains and Subdomains list in the Gateway service

Sample page 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0053)http://abc.siroe.com/mailclient/destin/?Cmd=navbar -->
<HTML XMLNS:WM><HEAD>
<META http-equiv=Content-Type content="text/html; CHARSET=utf-8">
<META http-equiv=Pragma content=no-cache>
<META http-equiv=Expires content=0><!--Copyright (c) 2000 Microsoft Corporation.
All rights reserved.--><!--CURRENT FILE== "IE5" "WIN32" navbar -->
<STYLE>WM\\:DROPMENU {
BEHAVIOR: url(http://abc.siroe.com/mailweb/controls/dropmenu.htc)
}
</STYLE>
<LINK href="destin_files/navbar.css" type=text/css rel=stylesheet>
<SCRIPT language=javascript>
var g_szUserBase= "http://abc.siroe.com/mailclient/destin"+"/";
var g_szFolder= ".";
var g_szVirtualRoot= "http://abc.siroe.com/mailweb";
var g_szImagePath= g_szVirtualRoot + "/img/";
</SCRIPT>
<SCRIPT src="/destin_files/navbar.js"></SCRIPT>
<META content="MSHTML 6.00.2600.0" name=GENERATOR></HEAD>
<BODY oncontextmenu=return(event.ctrlKey); onselectstart=return(false);
id=outbar_mainbody style="BACKGROUND-COLOR: appworkspace" leftMargin=0
topMargin=0 scroll=no>
<TABLE class=nbTableMain id=nbTableMain style="HEIGHT: 100%" cellSpacing=0
cols=1 cellPadding=0 rows="2">
<TBODY>
<TR>
<TD class=treeBrand>
```

```
<DIV class=treeOFLOW><IMG
style="PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 0px; PADDING-TOP: 0px"
src="/destin_files/logo-ie5.gif" border=0></DIV></TD></TR>
<TR height="100%">
<TD>
<TABLE class=nbTable cellSpacing=0 cols=1 cellPadding=0 rows="4">
<TBODY>
<TR>
<TD class=nbFlybar id=show_navbar onkeydown=flybar_keydown()
onclick=ToggleTab(this.id) tabIndex=0 noWrap>
<DIV class=treeOFLOW>Shortcuts</DIV></TD></TR>
<TR style="HEIGHT: 100%">
<TD id=idOutbarpane style="TEXT-ALIGN: center" vAlign=top><A
id=inbox
href="http://abc.siroe.com/mailclient/destin/Inbox/?Cmd=contents&amp;Page=1"
target=viewer alt="Go to inbox"><IMG class=nbImage alt="Go to inbox"
src="destin_files/navbar-inbox.gif"></A>
<DIV class=nbLabel>Inbox</DIV><BR><A id=calendar
href="http://abc.siroe.com/mailclient/destin/Calendar/?Cmd=contents"
target=viewer alt="Go to calendar"><IMG class=nbImage
alt="Go to calendar" src="destin_files/navbar-calendar.gif"></A>
<DIV class=nbLabel>Calendar</DIV><BR><A id=contacts
            href="http://abc.siroe.com/mailclient/destin/Contacts/?Cmd=contents"
target=viewer alt="Go to contacts"><IMG class=nbImage
alt="Go to contacts" src="destin_files/navbar-contacts.gif"></A>
<DIV class=nbLabel>Contacts</DIV><BR><A id=options
href="http://abc.siroe.com/mailclient/destin/?Cmd=options"
target=viewer alt="Go to options"><IMG class=nbImage
alt="Go to options" src="destin_files/navbar-options.gif"></A>
<DIV class=nbLabel>Options</DIV></TD></TR>
<TR style="HEIGHT: 1.5em">
<TD class=nbFlybar id=show_folders onkeydown=flybar_keydown()
onclick=ToggleTab(this.id) tabIndex=0 noWrap>
<DIV class=treeOFLOW>Folders</DIV></TD></TR>
<TR>
<TD class=nbTreeProgress id=treeProgress style="DISPLAY: none"
vAlign=top noWrap><SPAN id=idLoading
style="OVERFLOW: hidden">Loading...</SPAN>
</TD></TR></TBODY></TABLE></TD></TR></TBODY></TABLE>
</BODY></HTML>
```

Description

"Description" on page 85 shows the mapping between the sample ruleset and the case study.

TABLE 4–3 Mapping Between Sample Ruleset and Case Study

Page Content	Rule Applied	Rewriter Output	Description
`var g_szVirtualRoot= "http://abc.siroe.com/mailweb";`	<Variable name="URL"> g_szVirtualRoot </Variable>	`var g_szVirtualRoot= "http://gateway.sesta.com /http://abc.siroe.com/mailweb";`	`g_szVirtualRoot` is a variable whose value is a simple URL. This rule tells Rewriter to search for a variable `g_szVirtualRoot` of type URL. If such a variable exists in the web page, Rewriter converts this to an absolute URL, and prefixes the Gateway URL.
`src="/destin_files/ logo-ie5.gif"`	<Attribute name="src" />	`src="http://gateway.sesta.com/ http://abc.siroe.com/ destin_files/logo-ie5.gif`	`src` is the name of an attribute, and does not have any tag or valuePattern attached to it. This rule tells Rewriter to search for all attributes with the name `src`, and rewrite the value of that attribute.
`href="http://abc.siroe.com /mailclient/destin/Inbox/ ?Cmd=contents&Page=1"`	<Attribute name="href"/>	`href="http://gateway.sesta.com/ http://abc.siroe.com /mailclient/destin/ Inbox/?Cmd=contents&Page=1"`	`href` is the name of an attribute, and does not have any tag or valuePattern attached to it. This rule tells Rewriter to search for all attributes with the name `href`, and rewrite the value of that attribute.

Note – The order of priority for applying the ruleset is `hostname-subdomain-domain`.

For example, assume that you have the following entries in the Domain-based rulesets list:

```
sesta.com|ruleset1
eng.sesta.com|ruleset2
host1.eng.sesta.com|ruleset3
```

`ruleset3` is applied for all pages on `host1`.

`ruleset2` is applied for all pages in the `eng` subdomain, except for pages retrieved from `host1`.

`ruleset1` is applied for all pages in the `sesta.com` domain, except for pages retrieved from the `eng` subdomain, and from `host1`.

---

1. Click Save to complete.
2. Restart the Gateway from a terminal window:

```
./psadmin start-sra-instance –u amadmin – f  <password file> –N <profile name>– t  <gateway>
```

Ruleset for Outlook Web Access

Secure Remote Access server supports MS Exchange 2000 SP3 installation and MS Exchange 2003 of Outlook Web Access (OWA) on the Sun Java System Web Server and the IBM application server.

▼ To Configure the OWA Ruleset

1. **Log in to the Portal Server administration console as administrator.**
2. **Select the Secure Remote Access tab, and select the Gateway profile for which you want to set the attribute.**
3. **In the Map URIs to RuleSets field, enter the server name where Exchange 2000 is installed followed by the Exchange 2000 Service Pack 4 OWA ruleset.**

   For example:

   ```
   exchange.domain.com|exchange_2000sp3_owa_ruleset.
   ```

Using Public Folders

On the Exchange side Public Folders are configured to use NTLM Authorization. It needs to be changed to use HTTP Basic Authorization.

To do this, go to the Exchange server and select the Control Panel-->Administrative Tools, then open Internet Information Services.

Under Default Web Site there is a tab for Public Folders called Public. Right Click and select properties. Click on Directory Security Tab. Select "Edit.." on the Anonymous Access and Authentication control panel. Unselect everything else and select only Basic Authentication.

Mapping of 6.x RuleSet with 3.0

The following table lists the mapping of the Secure Remote Access server Rewriter rules with the previous releases of the Portal Server product.

TABLE 4–4 Mapping of Rules with SP3

Rewriter 6.0 DTD Element	Rewriter 3.0 List Box Name
Rules for HTML Content	
Attribute - URL	Rewrite HTML Attributes

TABLE 4–4 Mapping of Rules with SP3 *(Continued)*

Rewriter 6.0 DTD Element	Rewriter 3.0 List Box Name
Attribute - DJS	Rewrite HTML Attributes containing JavaScript
Form	Rewrite Form Input Tag List
Applet	Rewrite Applet/Object Parameter Values List
Rules for JavaScript Content	
Variable - URL	Rewrite JavaScript Variables in URL
Variable - EXPRESSION	Rewrite JavaScript Variables Function
Variable - DHTML	Rewrite JavaScript Variables in HTML
Variable - DJS	Rewrite JavaScript Variables in JavaScript
Variable - SYSTEM	Rewrite JavaScript System Variables
Function - URL	Rewrite JavaScript Function Parameters
Function - EXPRESSION	Rewrite JavaScript Function Parameters Function
Function - DHTML	Rewrite JavaScript Function Parameters in HTML
Function - DJS	Rewrite JavaScript Function Parameters In JavaScript
Rules for XML Content	
Attribute - URL	Rewrite Attribute value of XML Document
TagText	Rewrite Text data of XMl Document
Rules for CSS Content	
Rules are not required. By default, all URLs are translated	
Rules for WML Content	
No rules defined. WML is treated at HTML and HTML rules are applied.	
Rules for WMLScript Content	
No support for WML Script	

◆ ◆ ◆ CHAPTER 5

Policy Agent

You can limit the access of the web or portal sites by the SWA Gateway for a user by using an OpenSSO or Sun Access Manager policy agent.

Creating a Policy Agent

The following sections describe how to create a policy agent for your SWA Gateway.

Downloading and Installing a Policy Agent

You can download a J2EE Policy Agent for OpenSSO or a J2EE Policy Agent for Access Manager from Sun Downloads. To download a Policy Agent from Sun Downloads, select the View by Category tab, and select Access Manager under Identity Management.

Creating an Agent Profile and a Policy Agent

As described in the OpenSSO policy agent documents or Access Manager policy agent documents, you need to create an agent profile on `am.company.com` before you configure the agent.

Install the downloaded J2EE policy agent on the web container where you have deployed the Gateway. The following examples illustrate how to configure the J2EE policy agent to protect the Gateway. The examples assume that you have deployed the Gateway on a web container installed on a host with `gateway.company.com` as the FQDN (Fully Qualified Datastore Host) name, and that the protocol and port are `http` and `8080` respectively. An existing OpenSSO or Access Manager installation on `am.company.com` is assumed, with protocol `http` and port `80` respectively.

Modify the agent's `OpenSSOAgentConfiguration.properties` (assuming the agent configuration is local) or the `AMAgent.properties` file as follows.

1. In the `FILTER OPERATION MODE` section, add the following line:

   ```
   com.sun.identity.agents.config.filter.mode[gateway] = URL_POLICY
   ```

2. In the `LOGIN URL` section, modify the following line:

   ```
   com.sun.identity.agents.config.login.url[0] =
   http://am.company.com:80/opensso/UI/Login
   ```

   - For OpenSSO:

     ```
     com.sun.identity.agents.config.login.url[0] =
     http://gateway.company.com:8080/gateway/http://am.company.com:80/opensso/UI/Login
     ```

   - For Access Manager:

     ```
     com.sun.identity.agents.config.login.url[0] =
     http://gateway.company.com:8080/gateway/http://am.company.com:80/amserver/UI/Login
     ```

3. In the `NOT-ENFORCED URI PROCESSING PROPERTIES` section, modify the following line:

   ```
   com.sun.identity.agents.config.notenforced.uri[0] =
   ```

   - For OpenSSO:

     ```
     com.sun.identity.agents.config.notenforced.uri[0] =
     /gateway/http://am.company.com:80/opensso/*
     ```

   - For Access Manager:

     ```
     com.sun.identity.agents.config.notenforced.uri[0] =
     /gateway/http://am.company.com:80/amserver/*
     ```

4. In the `NOT-ENFORCED URI PROCESSING PROPERTIES` section, add the following line:

   ```
   com.sun.identity.agents.config.notenforced.uri[1]
   ```

   - For OpenSSO:

     ```
     com.sun.identity.agents.config.notenforced.uri[1] =
     /gateway/http://am.company.com/opensso/*
     ```

   - For Access Manager:

     ```
     com.sun.identity.agents.config.notenforced.uri[1] =
     /gateway/http://am.company.com/amserver/*
     ```

5. If you are using a 3.0 agent, add the following lines:

   - `com.sun.identity.agents.config.notenforced.uri[2] = /gateway/http://am.company.com:80/opensso/UI/Login?*`
   - `com.sun.identity.agents.config.notenforced.uri[3] = /gateway/http://am.company.com/opensso/UI/Login?*`

In the case of OpenSSO and a 3.0 agent, if the agent configuration is centralized on the OpenSSO Enterprise server, navigate to the J2EE Agent Properties page in the OpenSSO Enterprise Console, and perform the following steps. For the steps to navigate in OpenSSO Enterprise 8.0 Console to the J2EE Agent Properties, see http://docs.sun.com/app/docs/doc/820-4803/ghorc?a=view.

1. In the Global tab, add the following new value to the `Agent Filter Mode` property:
   a. Map Key: `gateway`
   b. Corresponding Map Value: `URL_POLICY`
2. Click `ALL` under Current Values and then click the *Remove* button.

   Restart the container where OpenSSO is installed.
3. In the OpenSSO Services tab, change the `OpenSSO Login URL` property from `http://`*am.company.com:80*`/opensso/UI/Login` to `http://`*gateway.company.com:8080*`/gateway/http://am.company.com:80/opensso/UI/Login`
4. In the Application tab, add the following new values to the `Not Enforced URIs` property:

   `/gateway/http://`*am.company.com:80*`/opensso/*`
   `/gateway/http://`*am.company.com:80*`/opensso/UI/Login?*`
   `/gateway/http://`*am.company.com*`/opensso/*`
   `/gateway/http://`*am.company.com*`/opensso/UI/Login?*`
5. Create a policy in *am.company.com* with rules to allow users to access at least the following resources:
   - `http://`*gateway.company.com:8080*`/gateway/`
   - `http://`*gateway.company.com:8080*`/gateway/index.jsp`
6. If you are installing the Policy Agent on GlassFish, make use of the workaround described in http://docs.sun.com/app/docs/doc/820-2539/gbbje?a=view

You can add more rules to open up access for the users. For more information about creating policies, see the policy agent documentation for OpenSSO at http://docs.sun.com/app/docs/coll/1767.1, or for Access Manager at http://docs.sun.com/app/docs/coll/1322.1.

Index

T

U

W

X