

HP OpenView Select Access

For the Windows®, Linux®, Solaris®, and HP-UX® Operating Systems

Software Version 6.2

Integration Paper for BEA WebLogic™ 9.1 Servers

Document Release Date: September 2006

Software Release Date: September 2006



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

© Copyright 2004-2006 Hewlett-Packard Development Company, L.P.

Trademark Notices

HP OpenView Select Access includes software developed by third parties. The software HP OpenView Select Access uses includes:

- Software developed by the Apache Software Foundation.
- Software developed by Claymore Systems, Inc.
- Cryptographic software written by Eric Young.
- Cryptographic software developed by The Cryptix Foundation Limited.
- cURL, Copyright 2000 Daniel Stenberg.
- JavaBeans Activation Framework version 1.0.1 Sun Microsystems, Inc.
- JavaMail, version 1.2 Sun Microsystems, Inc.
- JavaService software from Alexandria Software Consulting.
- JClass LiveTable, Copyright 2002 Sitraka Inc.
- The OpenSSL Project for use in the OpenSSL Toolkit.
- Protomatter Syslog, Copyright 1998-2000 Nate Sammons.
- SoapRMI, Copyright 2001 Extreme! Lab, Indiana University.
- WebLogic Server is a US trademark of BEA Systems, Inc.

For expanded copyright notices, see HP OpenView Select Access <install_path>/3rd_party_license directory.

Documentation Updates

This manual's title page contains the following identifying information:

- Software version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP OpenView Support web site at:

www.hp.com/managementsoftware/support

HP OpenView online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Contents

1	Understanding Your Select Access Integration	7
	Assumptions in this Document	7
	Integrating the Select Access and WebLogic	7
	Understanding the Key Components of this Integration	7
	The Select Access WebLogic Enforcer Plugin's Architecture	8
	The Authentication Provider	9
	The Identity Assertion Provider	10
	The Authorization Provider	10
	Supported Security Configuration Scenarios	10
2	Integration Tasks	15
	Deploying Select Access on a BEA WebLogic Host	16
	Getting the Required Files	16
	To install an Enforcer plugin	17
	To configure the plugin and create the XML bootstrap file	17
	Defining Select Access Classpaths	17
	Files You Need to Append	18
	To append Enforcer API JAR files to the classpath	18
	Storing WebLogic Connection Information	19
	About the bea_enforcer.properties File	20
	To create the bea_enforcer.properties file	20
	Making Select Access the WebLogic Security Provider	21
	Setting Select Access as WebLogic's Security Provider	21
	To set up Select Access as WebLogic's security providers	21
	Setting Default Security Constraints for WebLogic Resources	23
	Enabling Select Auth Authentication	23
	Modifying WebLogic Server Startup Scripts	23
	To configure the WebLogic Server to protect all resources	23
	Modifying Deployment Descriptors	24
	Configuring Select Access to Protect WebLogic Resources	25
	How Protocol Hierarchies Affect Resource Organization	27
	Constructing WebLogic resource hierarchies	29
	To bypass protocols	31
	When to Use Default Security Providers	32
	Setting Up Form-based Authentication	33
	Using Graphics on Login Pages	33
	Setting Up Form-based Authentication with Multiple DNS Domain SSO	34
	To set up SSO	34
	Setting Up Perimeter Authentication	36

To set up Perimeter authentication	38
To modify your HTTP 401 error page.....	38
Setting Up Authentication Via a Proxy Server	38
To set up authentication via a proxy	39
Setting Up Certificate Authentication	39
Setting Up Multiple Authentication Methods	39
Setting Up Personalization	40
How it Works in Select Access' WebLogic Enforcer Plugin	40
To extract personalization data	40
Integration with the WebLogic Portal.	42
Configure Resources Required During Server Startup	44
To determine resource denials	45
Configure Security Policy and User Profiles in the WebLogic Administration Portal	45

1 Understanding Your Select Access Integration

Select Access is an integral part HP's comprehensive Identity Management suite. It delivers a full solution for complex access management across the enterprise. Select Access:

- Automates access control and user life-cycle management
- Extends the enterprise through federation
- Delegates management to business owners and the end users themselves

Along with robust workflow, user self-service, reporting, and delegated administration capabilities, Select Access is the most comprehensive access control system available. Select Access simplifies your ability to secure user access to BEA WebLogic 9.1 Server™ resources.

Assumptions in this Document

This document assumes the following:

- That you have BEA WebLogic 9.1 installed and running on your network
- That you understand the features and functions of Select Access
- That you have updated Select Access 6.0 with Engineering Patch 1B
- That you have a working knowledge of Select Access and LDAP 3.0-compliant directory servers

Integrating the Select Access and WebLogic

To understand the integration between Select Access and the WebLogic Server, you need to understand where critical integration points lie.

Understanding the Key Components of this Integration

WebLogic's Security Service architecture consists of three major components:

- the WebLogic Security Framework
- the Security Service Provider Interfaces (SSPIs)
- the WebLogic Security Providers

Select Access' WebLogic Enforcer plugin protects WebLogic Server resources by adding three Select Access Security Providers to WebLogic Server. In so doing, Select Access' WebLogic Enforcer plugin assigns all authentication and authorization responsibilities to the Policy Validator. [Figure 1](#) shows a high-level view of WebLogic Security Service Architecture.

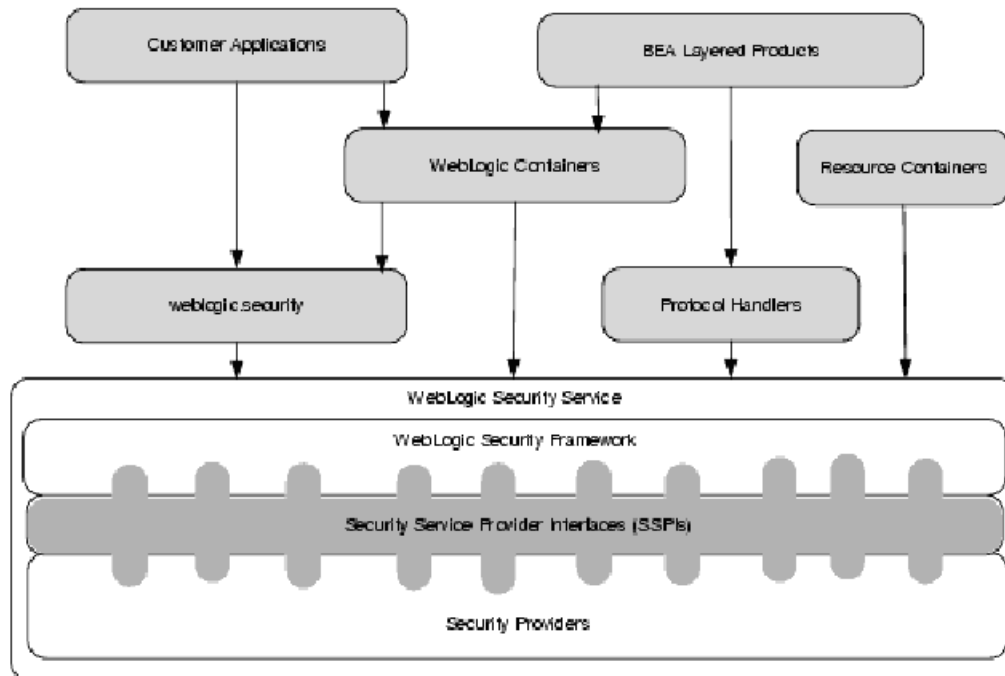


Figure 1 WebLogic Security Service Architecture

The WebLogic Enforcer plugin differs in structure from other Enforcer plugins available with Select Access: it is implemented via independent JAR files (`SASecurityProvidersWL9.jar` and `SAPrincipal.jar`), but it is also written using two APIs:

- WebLogic’s Security Service API: To create a pluggable security provider on the WebLogic server for the realm you designate.
- Select Access’ Enforcer API: To create security providers that delegate authentication and authorization to Select Access Policy Validator.

Because the implementation and deployment of Select Access’ WebLogic Enforcer plugin is unique to this integration, the method for configuring password authentication, Single Sign-On (SSO), and personalization differs from other Enforcer plugins. For details, see [Chapter 2, Integration Tasks](#).

The Select Access WebLogic Enforcer Plugin’s Architecture

Select Access’ WebLogic Enforcer plugin includes three types of security providers. These three security providers act as the agent for Select Access on the WebLogic Server. The actual authentication and authorization is still done by a Policy Validator. Security providers use the Enforcer API to communicate with a Policy Validator and use WebLogic SSPIs to communicate with the WebLogic Security Framework. These providers are described below:

- Authentication Provider: Checks user’s credentials and populates the principal object to the WebLogic Server. For details, see [The Authentication Provider](#) on page 9.
- Identity Assertion Provider: Performs perimeter authentication based on Select Access tokens (known as nonces). For details, see [The Identity Assertion Provider](#) on page 10.
- Authorization Provider: Makes access decisions based on policy.

The plugin also includes a servlet:

- Login Servlet: Gathers user login information from either a:
 - login form
 - cookie
 - URL

It then provides that information to Authentication Providers and Identity Assertion Providers. The Login Servlet is also used to do redirects for MD-SSO.

Figure 2 illustrates the high-level architecture of Select Access' WebLogic Enforcer plugin.

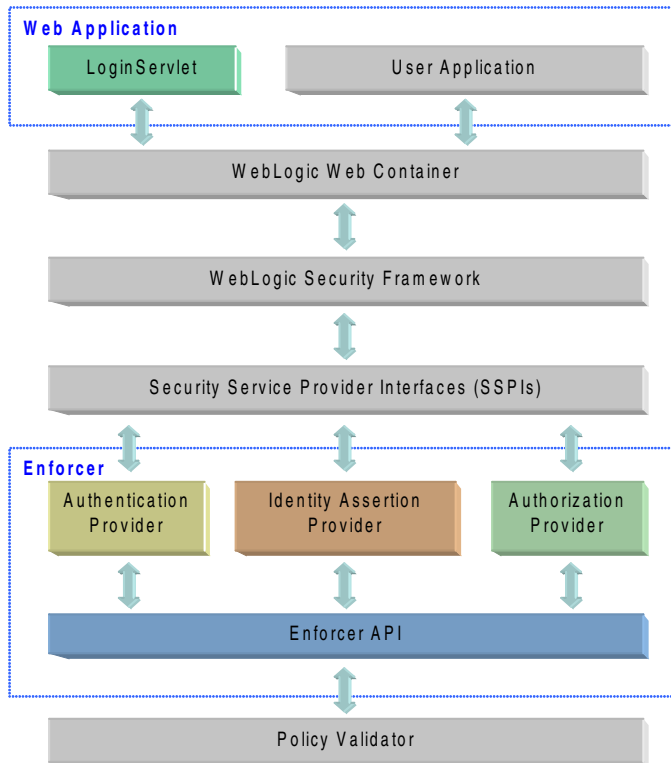


Figure 2 Select Access' WebLogic Enforcer Plugin Architecture

The Authentication Provider

The Enforcer plugin's Authentication Provider component delegates the authentication to Policy Validator. It supports six types of authentication: Password, SecurID, RADIUS, certificate, NTLM and Kerberos. Depending on the type of client, the authentication implementation can vary:

- Fat-clients (standalone Java clients) can use Password, SecurID, NTLM, and Kerberos authentication.
- Thin-clients (browser clients). BEA WebLogic Server supports HTTP Basic Authentication or Form-based Authentication to authenticate to the Select Access Authentication Provider. Only Form-based Authentication supports all of the Select

Access Authentication methods (password, certificate, SecurID, Radius, NTLM, and Kerberos authentication). You can configure the authentication method to be used through the web application's deployment descriptor configuration.

For details, see [Setting Up Form-based Authentication with Multiple DNS Domain SSO](#) on page 34.

The Identity Assertion Provider

The Identity Assertion Provider is the component that performs Perimeter Authentication by using a nonce: a unique token required by the Select Access system. Perimeter Authentication is the authentication method that allows users or system processes to assert their identity using tokens, and therefore, supports single sign-on (SSO). The Identity Assertion Provider can authenticate users by reading the token set by a different system protected by Select Access, and stored in the cookie.

The Authorization Provider

The Authorization Provider is the component that delegates the authorization to the Policy Validator. The Policy Validator can control access to the following WebLogic Server resources:

- URL Resources (HTML, JSP, Servlet)
- EJB Resources
- JNDI Resources
- JDBC Resources
- JMS Resources
- Application Resources
- Server Resources
- Administration Resources
- Web Services Resources

Supported Security Configuration Scenarios

Table 1 lists the security scenarios you can configure with Select Access' WebLogic Enforcer plugin. These options vary in the degree of difficulty required to set it up.

Table 1 Supported Authentication Scenarios

Type	Difficulty	Use this When	What is Required
HTTP Basic	Simple	<ul style="list-style-type: none"> • You do not need Select Access to control session timeouts. Choose another method if this is required. • You want users signing in from a WebLogic Server to also log into other systems protected by Select Access in the same DNS domain. <p><i>Note:</i> The reverse is not possible</p>	<ul style="list-style-type: none"> • Configure Select Access' Authentication and Authorization Providers. • No configuration for web application required.

Table 1 Supported Authentication Scenarios (cont'd)

Type	Difficulty	Use this When	What is Required
Form	Moderate	<ul style="list-style-type: none">You want a custom login page for your applications.You want users signing in from a WebLogic Server to also log into other systems protected by Select Access in the same DNS domain. <p>Note: The reverse is not possible</p>	<ul style="list-style-type: none">Configure Select Access' Authentication and Authorization Providers.Configure the web application to use Form Authentication.<ul style="list-style-type: none">Create two pages: login and login error.Modify the application's deployment descriptor to indicate the FORM authentication method. <p>Note: You cannot globally configure all applications on the WebLogic Server.</p> <p>For details, see Setting Up Form-based Authentication on page 33.</p>

Table 1 Supported Authentication Scenarios (cont'd)

Type	Difficulty	Use this When	What is Required
Form with multiple DNS domain SSO ^a	Moderate	<ul style="list-style-type: none"> • You want a custom login page for your applications. • You want users signing in from a WebLogic Server to also log into other systems protected by Select Access. The other systems can be in different DNS domains. • You want users signing in from other systems protected by Select Access to also log into the WebLogic Server. The other systems can be in different DNS domains. 	<ul style="list-style-type: none"> • Configure all three Select Access Security Providers. • Configure the web application to use Form Authentication. This requires that you: <ul style="list-style-type: none"> — Modify the application's deployment descriptor to indicate the FORM authentication method. — Configure Select Access' Login Servlet as the login page for the application. — Configure authentication server and login form by using Select Access Policy Builder. <p>Note: You cannot globally configure all applications on the WebLogic Server.</p> <p>For details, see Setting Up Form-based Authentication with Multiple DNS Domain SSO on page 34.</p>

Table 1 Supported Authentication Scenarios (cont'd)

Type	Difficulty	Use this When	What is Required
Perimeter	Moderate	<ul style="list-style-type: none"> You do not plan to let users log in from the WebLogic Server. For example, you have a dedicated login server and all users must log in from that server. You want users signing in from other systems protected by Select Access to also log into the WebLogic Server. However, users cannot use Forms or HTTP Basic Authentication to log into the WebLogic Server directly, due to a limitation of WebLogic Server's web container. Do not require automatic redirects. When the session timeout token expires, the browser instead displays an unauthorized message to user. However, you can configure the WebLogic Server's unauthorized error page (401) to redirect user to your login server. 	<ul style="list-style-type: none"> Configure all three Select Access Security Providers. Modify the application's deployment descriptor to indicate the CLIENT-CERT authentication method. <p>Warning: If you setup FORM or BASIC authentication in your web application, WebLogic Server does not perform perimeter authentication, even if you configured Select Access' Identity Assertion Provider.</p> <p>For details, see Setting Up Perimeter Authentication on page 36.</p>
Proxy servers		<ul style="list-style-type: none"> Want the Proxy server to authenticate users. Want full protection of all the resources supported by Select Access' WebLogic Server Enforcer plugin, not only the URL resources protected by the Proxy server. 	<ul style="list-style-type: none"> Install and configure an appropriate Enforcer plugin on the Proxy server. Configure all three Select Access Security Providers. Modify the application's deployment descriptor to indicate the CLIENT-CERT authentication method. <p>Warning: If you setup FORM or BASIC authentication in your web application, WebLogic Server does not perform perimeter authentication, even if you configured Select Access' Identity Assertion Provider.</p> <p>For details, see Setting Up Authentication Via a Proxy Server on page 38.</p>

a. HP recommends this scenario.

2 Integration Tasks

To ensure Select Access and the WebLogic Server function properly as a unit, you must follow a specific series of steps to integrate them correctly. [Table 1](#) summarizes the steps you need to perform, to configure and delegate all authentication and authorization responsibilities to Select Access.

- Due to requirements with Select Access' WebLogic Enforcer plugin, you must update your deployment of Select Access 6.0 with Engineer Patch 1B. Please make sure you have installed the patch before starting any integration tasks.
- You can still use WebLogic's native security mechanisms, however HP recommends that you do so to support WebLogic's internal functions only. See [When to Use Default Security Providers](#) on page 32 for details.

Table 1 Integration Overview

Integration Tasks	Details
1 Install Select Access' WebLogic Enforcer plugin files on the same host as the WebLogic Server.	Deploying Select Access on a BEA WebLogic Host on page 16
2 Define the classpath for Enforcer JAR files. The classes in these JAR files are used by the BEA WebLogic Server.	Defining Select Access Classpaths on page 17
3 Create a properties file for the Enforcer plugin.	Storing WebLogic Connection Information on page 19
4 Configure the BEA WebLogic Server to use Select Access as its Security Provider.	Making Select Access the WebLogic Security Provider on page 21
5 Set default security constraints for WebLogic resources so that they are protected as they are added to the Policy Builder.	Setting Default Security Constraints for WebLogic Resources on page 23

Table 1 Integration Overview (cont'd)

Integration Tasks	Details
6 Configure the Policy Matrix in the Policy Builder so it displays the correct combination of users, resources, and services.	Configuring Select Access to Protect WebLogic Resources on page 25
7 Choose and configure your preferred authentication method.	Setting Up Form-based Authentication on page 33 OR Setting Up Form-based Authentication with Multiple DNS Domain SSO on page 34 OR Setting Up Perimeter Authentication on page 36 OR Setting Up Authentication Via a Proxy Server on page 38 OR Setting Up Certificate Authentication on page 39 OR Setting Up Multiple Authentication Methods on page 39
8 Set up personalization, if required.	Setting Up Personalization on page 40

Deploying Select Access on a BEA WebLogic Host

You can install Select Access components across any combination of host computers. However, to ensure that your WebLogic Server has the right components and files required to make Select Access its Security Provider, you need to install one Select Access Enforcer plugin on WebLogic's host machine.




This can be any of Select Access' Enforcer plugin. It is not, however, the Select Access WebLogic Enforcer plugin. You will be installing this plugin manually, later in this document. These WebLogic-specific files are not installed as part of Select Access 6.2.

Getting the Required Files

At minimum, HP recommends you install and configure one of the Enforcer plugins available with the Select Access 6.2 installer on the WebLogic host. This ensures that the required Enforcer API files are installed locally. Upon completion, you need to create the `enforcer_bea.xml` bootstrap file. This file contains the configuration parameters that are required by the Enforcer API classes that are part of Select Access' WebLogic Enforcer plugin JAR files.

To install an Enforcer plugin

- 1 Run the Select Access 6.2 installer.
- 2 Follow the instructions described in [Chapter 3, Installing Select Access](#), in the *HP OpenView Select Access 6.2 Installation Guide*.
- 3 When you have reached the **Choose HP OpenView Select Access Components** screen, select one of the following check boxes to install all of the correct Enforcer API files:
 - Sun/Netscape/iPlanet Enforcer plugin
 - IIS Enforcer plugin
 - Apache 2 Enforcer plugin

 You do not need to have web server software installed on WebLogic's host machine to install and configure an Enforcer plugin.
- 4 Click **Next** and finish the installation as required.

To configure the plugin and create the XML bootstrap file

- 1 Run the Setup Tool as described in [Chapter 4, Configuring Select Access](#), in the *HP OpenView Select Access 6.2 Installation Guide*.
- 2 Click **Next** until you reach the **Generic Enforcer plugin** setup screen. This wizard creates the XML bootstrap file Select Access' WebLogic Enforcer plugin requires.
- 3 Click **Browse** and select the folder to where the XML file will be created.
- 4 Name and specify the path for the file you are about to generate. HP recommends that you save the file to the same folder as the Web Logic Domain. For example, `<WL_Domain_path>/enforcer_bea.xml`. This file name must match the path file name you configure in your properties file. For details, see [About the bea_enforcer.properties File](#) on page 20.
- 5 Click **Configure** and configure the Enforcer plugin as required. For Select Access' WebLogic Enforcer plugin, a **Typical** configuration is suitable for most deployments. For details, see [Configuring the Enforcer Plugin](#) on page 129, in the *HP OpenView Select Access 6.2 Installation Guide*.
- 6 Click **Finish** to exit the Setup wizard and commit the configuration to the XML file.

Defining Select Access Classpaths

Because the WebLogic Server uses the Enforcer API classes in Select Access' WebLogic Enforcer plugin JAR files, you need to append the required Select Access files to the CLASSPATH of WebLogic Server.

Typically, you define CLASSPATHs in the WebLogic Server's startup script. By default the name of startup script is `startWebLogic.cmd` and the name of closing script is `stopWebLogic.cmd`, both of which are stored in the `/bin` directory, under the WebLogic Server Domain directory.

Files You Need to Append

You need to append the following JAR files to the CLASSPATH section of the WebLogic Server's startup script and closing script:

- `SAPrincipal.jar`: Contains the `SAPrincipalImpl` class that extends the `WLSUser` interface. It allows WebLogic to use its default security providers with Select Access' Enforcer plugin in the same security realm. The `SAPrincipalImpl` class accomplishes this by representing Select Access users as native users in the WebLogic system.
- `EnforcerAPI.jar`: Contains all the classes used to create an Enforcer plugin.



These files are downloadable from the HP OpenView Select Access support site.

To append Enforcer API JAR files to the classpath

- 1 Create a new folder for holding all the JAR files Select Access' WebLogic Enforcer plugin requires:

```
<WebLogic_install_path>/server/lib/sa
```

- 2 Copy the JAR files listed in [Table 2](#) into the folder you just created.

Table 2 Files You Need

Filename	Where to Get it
<code>servletfilter.war</code> ^a	Extract all files from the <code>servletfilter.war</code> file.
<code>SAPrincipal.jar</code>	Download the <code>SAPrincipal.jar</code> from the HP OpenView Select Access support site to the same folder as the <code>EnforcerAPI.jar</code> .
<code>jdom.jar</code>	Download the latest version (jdom version 1.0) from http://www.jdom.org .
<code>jce1_2_2.jar</code> ^b <code>local_policy.jar</code> ^b <code>US_export_policy.jar</code> ^b	<ol style="list-style-type: none">1 Download and extract the Java Cryptography Extension (JCE) 1.2.2 file (<code>jce-1_2_2.zip</code>) from java.sun.com.2 Browse to the <code>jce1.2.2\lib</code> folder.

a. `servletfilter.war` can be found under the `servlet` directory on the SA install CDs.

`<SA_install_path>` is where you installed Select Access, e.g. `C:\Program Files\HP OpenView\Select Access`.

b. Only required for JRE 1.3. HP recommends you use JRE 1.4.

- 3 Open your startup script for the WebLogic Server. By default the name of startup script is `\bin\startWebLogic.cmd`, which is stored in the domain directory you configured with the WebLogic Configuration wizard.
- 4 Create an `SA_LIB` that points to the Select Access folder you created in [Step 1](#). Using the example provided in that step, the variable added to the startup script is:

```
set SA_LIB=C:\bea\weblogic91\server\lib\sa
```

- 5 Create an `SA_CLASSPATH` variable that includes all the JAR files listed in [Table 2](#). This defines the correct classpaths to the required Select Access JAR files described in [Table 2](#). Without these paths, the WebLogic Server would not know where to find the files for the Enforcer plugin. [Sample of the SA_CLASSPATH variable](#) shows the contents of this new variable.

- ▶ If you want to condense the path, use the `SA_LIB` variable you created in [Step 4](#).
- ▶ Depending on your platform, the separators you should use will vary: on UNIX, use a colon (:), on Windows, use a semicolon (;).

- 6 Locate the `CLASSPATH` variable and append `SA_CLASSPATH` variable as another value for the `CLASSPATH` variable. For example:

```
set CLASSPATH=%CLASSPATH%;%SA_CLASSPATH%
```

- 7 Save the changes to this file.

- 8 Open your closing script for the WebLogic Server. By default the name of startup script is `\bin\stopWebLogic.cmd`, which is stored in the domain directory you configured with the WebLogic Configuration Wizard.

- 9 Repeat Steps 4 to 7 for the closing script.

- ▶ If a different variable is used for `CLASSPATH`, as in the example below which uses `%WEBLOGIC_CLASSPATH%`, use that variable in place of `CLASSPATH`:

```
%JAVA_HOME%\bin\java -cp %WEBLOGIC_CLASSPATH% weblogic.Admin  
FORCESHUTDOWN
```

Sample of the SA_CLASSPATH variable

The following is a sample of the `SA_CLASSPATH` variable:

```
set SA_LIB=/opt/bea/weblogic91/server/lib/sa  
set SA_CLASSPATH=${SA_LIB}:${SA_LIB}/bcprov-jdk14.jar:  
${SA_LIB}/SAPrincipal.jar:${SA_LIB}/castor-0.9.3.19-xml.jar:  
${SA_LIB}/EnforcerAPI.jar:${SA_LIB}/jdom.jar:  
${SA_LIB}/ldapjdk.jar:${SA_LIB}/protomatter.jar:  
${SA_LIB}/msgsrresources.jar:${SA_LIB}/shared.jar:${SA_LIB}/  
xercesImpl.jar:  
${SA_LIB}/xml.jar:${SA_LIB}/jakarta-oro-2_0.jar:${SA_LIB}/mail.jar:  
${SA_LIB}/xml-apis.jar:${SA_LIB}/US_export_policy.jar:  
${SA_LIB}/jce1_2_2.jar:${SA_LIB}/local_policy.jar  
set CLASSPATH=${CLASSPATH}:${SA_CLASSPATH}
```

Storing WebLogic Connection Information

In order for the Enforcer plugin to send valid queries to the Policy Validator for WebLogic resources, it must know the service name and root resource path defined for the protected WebLogic Server. The Enforcer plugin reads this information from a configuration file called `bea_enforcer.properties`.

About the `bea_enforcer.properties` File

This file is not native to Select Access nor to the WebLogic Server. In order to use Select Access with WebLogic, you must create this configuration file and save it to the domain directory you configured with the WebLogic Configuration wizard.



The service name you define as a parameter in this file must exactly match the host name and port that you configured for your WebLogic services. You will eventually add these services to the Policy Matrix. See [Configuring Select Access to Protect WebLogic Resources](#) on page 25 for information about adding WebLogic services with the Policy Builder.

To create the `bea_enforcer.properties` file

- 1 In a text editor create a new file.
- 2 Add the parameters listed in [Table 3](#) to the file.
- 3 Save the file. When you are finished, your properties file should look similar to the one shown in [Sample `bea_enforcer.properties` file](#).

Table 3 Properties You Need to Add

Property Name	Description
<code>EnforcerAPIConfigFile</code>	This file name must match the file name you defined for your XML configuration file. For details, see To configure the plugin and create the XML bootstrap file on page 17.
<code>Service</code>	Identifies where the WebLogic Server is located using the host name for the computer on which the WebLogic Server is located and the a port number. This parameter uses the following syntax: <code>Service=<hostname>:<port></code>
<code>Resource</code>	Identifies the root directory that all queries to the Policy Validator start with. This parameter uses the following syntax: <code>Resource="<root_path>"</code> Note: Typical deployments often use a value of <code>/</code> as the root path.
<code>SecurityRealm</code>	The security realm used by Perimeter Authentication. Set it to the Select Access' Security Realm.

Sample `bea_enforcer.properties` file

The following is a sample of the `enforcer.properties` file:

```
EnforcerAPIConfigFile=enforcer_bea.xml
Service=myBEAas:7001
Resource=/
SecurityRealm=SARealm
```

Making Select Access the WebLogic Security Provider

A critical integration step is to make Select Access the Security Provider for the WebLogic Server. This step ensures that Select Access components, not WebLogic's native mechanisms, protect the application server and its available resources. Setting up Select Access as the Security Provider requires that you follow the steps described in [Table 4](#).

Table 4 Making Select Access the Security Provider

Setup Task	Configuration Details
<p>Step 1: Copy the Java archive file that contains the Security Providers to the required location on the WebLogic Server host computer.</p>	<ol style="list-style-type: none"> 1 Stop the WebLogic Server. 2 Copy <code>SASecurityProvidersWL9.jar</code> that you downloaded from the HP OpenView Select Access support site. 3 Move it to the following folder: <code><WebLogic_install_path>/server/lib/mbeatypes</code> 4 Restart the server.
<p>Step 2: Set up the WebLogic Server to use Select Access as its Security Provider.</p> <p>Note: For specific procedural details for any of the tasks listed for this step, refer to the WebLogic Server's documentation.</p>	<ol style="list-style-type: none"> 1 Open the WebLogic Server Console in a web browser of your choice. 2 Create and configure a new security realm for Select Access by using default settings provided in the Create a New Realm page. The only unique property you want to manually configure is the realm name, for example <code>SArealm</code>. 3 Click the realm you created in Step 2, and create and configure its Security Providers. You must set up these according to HP's recommendations. For details, see Setting Select Access as WebLogic's Security Provider on page 21. 4 Because you can have multiple security realms in a WebLogic Server domain, but only one set as the default (active) realm, ensure the Select Access realm you created in Step 2 is the default security realm of your domain.
<p>Step 3: Restart the WebLogic Server.</p>	<p>WebLogic Server's documentation</p>

Setting Select Access as WebLogic's Security Provider

You must define Select Access as the WebLogic Server's Security Provider. You can still use other security providers, but they should only support WebLogic's internal functions.

To set up Select Access as WebLogic's security providers

- 1 In the `<realm_name>` page for the realm you created in Step 2 of [Table 4](#), click the **Providers** tab.

- 2 In the **Adjudication** tab, create and configure an **Adjudication Provider**.
- 3 In the **Auditing** tab, optionally create and configure an **Auditing Provider**.
- 4 In the **Authentication** tab, create and configure the following **Authentication Providers** in the order outlined in [Table 5](#).

Table 5 Required Order of Authentication Providers

Provider Name	Required?	Configuration Details
1 SAAuthenticator	yes	In the Common tab, set the Control Flag set to <code>OPTIONAL</code> .
2 Default Authenticator	yes	In the Common tab, set the Control Flag set to <code>OPTIONAL</code> .
3 SAIdentity Asserter ^a	for Perimeter Authentication only	In the Provider Specific tab, make <code>PolicyUser</code> an active token by moving it from the Available column to the Chosen column.
4 DefaultIdentity Asserter ^a	for Perimeter Authentication only	In the Provider Specific tab, make <code>AuthenticatedUser</code> an active token by moving it from the Available column to the Chosen column.

a. Required for Form-based Authentication with SSO, perimeter authentication, and authentication via proxy only.

- 5 In the **Authorization** tab, create and configure `SAAuthorizer` and `Default Authorizer`, by using WebLogic's default values.
- 6 In the **Credential Mapping** tab, create and configure a `Default Credential Mapper`, by using WebLogic's default values.
- 7 In the **Role Mapping** tab, create and configure a `Default Role Mapper`, by using WebLogic's default values.

Setting Default Security Constraints for WebLogic Resources

Due to the design of the J2EE architecture, WebLogic resources cannot use Select Auth for authentication by default. While you can still enable Select Auth in the Policy Builder for each resource, the authentication mechanisms will have no effect. This restriction exists because security for each J2EE resource is controlled by a deployment descriptor XML file that is resource-specific. However, this descriptor file does not initially include default security constraints.

Enabling Select Auth Authentication

To enable Select Auth authentication, you must manually set these security constraints for each resource, before adding the resource in question to the Policy Builder's Resources Tree. Set security constraints for WebLogic resources such that they are protected by default as you add them to the Policy Builder. There are two methods you can use:

- WebLogic Server startup scripts. HP recommends you use this method. For details, see [Modifying WebLogic Server Startup Scripts](#) on page 23.
- Deployment descriptors for each resource. For details, see [Modifying Deployment Descriptors](#) on page 24.

Modifying WebLogic Server Startup Scripts

This method allows you to configure the WebLogic Server to protect all WebLogic resources by default. Of the two methods available to you, this method is more secure than the modification of deployment descriptors. In this case, Select Access passes a flag to the WebLogic Server upon the application server's startup. This flag forces it to protect all resources, irrespective of any security constraint information specified in the deployment descriptors.



Users can be transferred from the WebLogic internal user store to an LDAP directory using the Migrate functionality in the WebLogic Admin Console.

To configure the WebLogic Server to protect all resources

- 1 Open the `<WLDomain_path>/bin/startWebLogic.cmd` file.
- 2 Do one of the following:
 - If you have a `JAVA_OPTIONS` variable, append the following parameter to the end of it:
`-Dweblogic.security.fullyDelegateAuthorization=true`
 - If you do not have a `JAVA_OPTIONS` variable, create this variable and set the required parameter for it:
`set JAVA_OPTIONS=
-Dweblogic.security.fullyDelegateAuthorization=true`
- 3 Ensure the `JAVA_OPTIONS` variable appears in the line that starts WebLogic Server. This line is automatically included if you used the Configuration wizard to setup WebLogic. Otherwise, if you have manually created your file, you may need to append this line.

Sample Startup line shows this variable in bold. All other variables vary depending on your deployment of WebLogic.

- 4 Save the file and restart your server. By default, all WebLogic resources are protected by Select Access with these changes.

▶ You must set the corresponding access policy for each user in the Policy Builder. See [Configuring Select Access to Protect WebLogic Resources](#) on page 25 for details.

Sample Startup line

The example below shows how to use the `JAVA_OPTIONS` variable in the startup line:

```
%JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%  
-Dweblogic.Name=%SERVER_NAME% -Dweblogic.management.username=%WLS_USER%  
-Dweblogic.management.password=%WLS_PW%  
-Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE%  
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy"  
weblogic.Server
```

Modifying Deployment Descriptors

Deployment descriptors are XML documents which define how a resource is used. By adding a security constraint to the deployment descriptor of a resource, you can control how the resource is accessed.

▶ The deployment descriptors must be modified with security information as defined in the J2EE specification. HP does not recommend this option as it is less secure than the method described in the previous section. Therefore, modifying deployment descriptors is beyond the scope of this document.

Configuring Select Access to Protect WebLogic Resources

This integration currently cannot support network discoveries on a WebLogic Server. Until this feature is fully implemented, you need to manually build the Resources Tree in the Policy Builder. [Table 6](#) describes the steps required to protect WebLogic resources.

Table 6 Configuring Select Access

Step	Configuration Details
<p>Step 1: Manually add the service you want to protect using Select Access.</p>	<ol style="list-style-type: none"> 1 Right-click a folder or the root of the Resources Tree. 2 Click New → Service. The New Service dialog box appears. 3 Name the resources and configure the service depending on whether or not the service is specific to a single server. For details see either: <ul style="list-style-type: none"> • Entering Information for a Non-Server-Specific Resource Service on page 43 in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>. • Entering Information for a Server-Specific Network Resource Service on page 43 in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>. <p>Note: The host name and port number you specify must exactly match the host name and port number configured in the <code>bea_enforcer.properties</code> file. For more information on this file, see To create the bea_enforcer.properties file on page 20.</p> <p>Note: Setup the proper protocol for the service according to the resource type you want to protect. You may need to setup mutiple service nodes as each has a different protocol. A service node with protocol “WebLogic” is mandatory, because that service node will be used for authentication. For a complete protocol list please refer to Table 7 Resource Hierarchies on page 29.</p> 4 Click OK.
<p>Step 2: Add the WebLogic resources you would like protected below the service entry you created in Step 1.</p> <p>Caution: Before adding resources, first consider the resource structure of the WebLogic system; the resource architecture varies depending on the protocol deployed. Each resource type has a hierarchy. For more details, see How Protocol Hierarchies Affect Resource Organization on page 27.</p>	<ol style="list-style-type: none"> 1 Right-click the WebLogic service you just created 2 Click New → Resource. The New Resource dialog box appears. 3 Name the resource. 4 Click OK. <p>For details, see To create a new network resource or edit an existing one on page 49 in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>.</p>

Table 6 Configuring Select Access (cont'd)

Step	Configuration Details
<p>Step 3: Enable Select Auth and choose an Authentication server for the service you created in Step 1. Select Access' WebLogic Enforcer plugin supports six types of authentication servers:</p> <ul style="list-style-type: none"> • password • SecurID • RADIUS • NTLM • Kerberos • Certificate <p>Note: You must enable Select Auth for ALL the service you created and make sure they are all configured to use the same authentication servers.</p>	<ol style="list-style-type: none"> 1 On the Select Auth column, enable Select Auth in the corresponding cell for the service. The Authentication Properties dialog box appears. For details on Select Auth, see Setting a Select Auth Policy on page 82 in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>. 2 Click the Authentication tab and do the following: <ol style="list-style-type: none"> a If you have not already configured an authentication server, click Add. The Available Authentication Servers dialog box appears allowing you to configure the server as required. For details on authentication servers, see Chapter 7, <i>Setting up authentication servers</i>, in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>. b If you have already configured an Authentication server, select one or more server names from the list of Available Servers and click Add. The server names appear in the Selected Servers window. To reorder the servers, select a server in the Selected Servers list and click the up arrow or down arrow buttons. c Click OK. 3 Optional. To supply a meaningful user name to the WebLogic Server and your applications, you need to configure an LDAP attribute from a user's entry that Select Access sets as the user name. Select Access also uses this attribute to set the result of <code>HttpServletRequest.getRemoteUser()</code> method. To define an attribute that will act as a meaningful user name: <ol style="list-style-type: none"> a Click the Personalization tab. b Create a new environment variable, <code>UID</code>. c Choose an LDAP attribute from the dropdown list. For example, <code>uid</code> or <code>userid</code>. <p>This procedure should be performed on the service node which has the 'weblogic' protocol.</p> <p>If you choose not to define a custom user name, Select Access will use the user's DN as user name by default.</p> 4 Configure the rest of personalization data as your specific deployment requires. Make sure you configure the personalization data on the proper Resources Tree, according to your resource type. For details on enabling personalization, see Chapter 5, Authentication Basics: Select Auth & Personalization, in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>. For details on specific issues with the Select Access-WebLogic integration, see Setting Up Personalization on page 40.

Table 6 Configuring Select Access (cont'd)

Step	Configuration Details
<p>Step 4: Assign access policies to your users and WebLogic resource combinations. Select Access can handle all authorization requirements you have, when you use any combination of allow/deny/conditional policies.</p> <p>Note: There is one exception. For details, see When to Use Default Security Providers on page 32.</p>	<p>For details, see Chapter 7, Controlling Network Access, and Chapter 8, Creating Conditional Access Rules with the Rule Builder, in the <i>HP OpenView Select Access 6.2 Policy Builder Guide</i>.</p>

How Protocol Hierarchies Affect Resource Organization

In order to add a sub-resource to an existing resource, you must understand how the resource hierarchy is constructed for each protocol. Without knowing the protocol, you cannot add a valid sub-resource. Therefore, unless you give some forethought to how you intend to organize your resources in the Resources Tree, your ability to see what protocol a resource uses limited to displaying its properties.

To easily identify which protocol a resource uses, HP recommends that you group resources of a similar protocol into well-organized folders. Figure 3 illustrates how WebLogic resources might be organized in the Resources Tree.

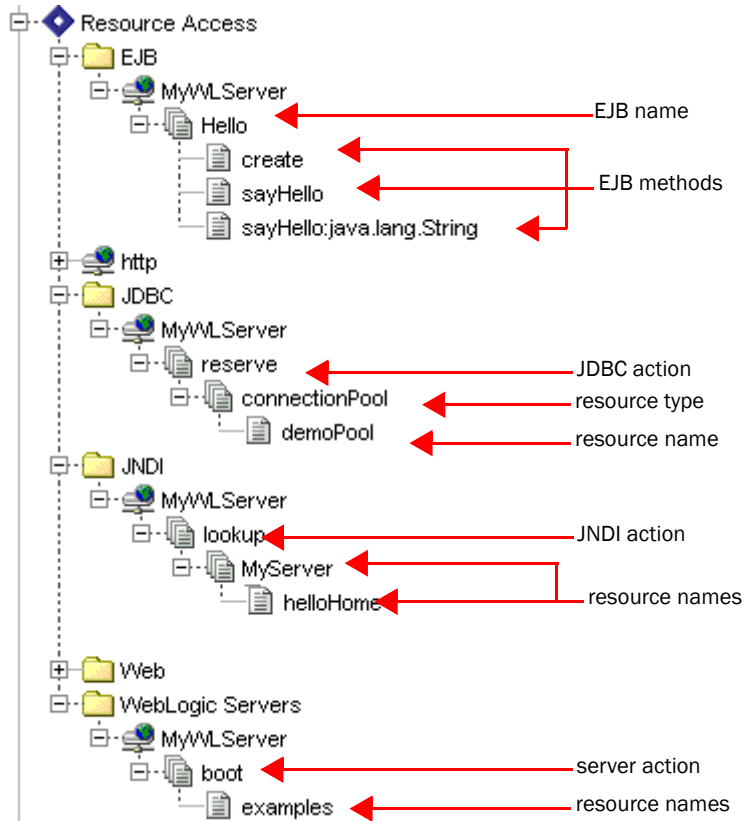


Figure 3 WebLogic Resources in the Policy Matrix

For a description of each available protocol in the WebLogic Server, see [Constructing WebLogic resource hierarchies](#) on page 29 for a detailed description of each protocol.

Constructing WebLogic resource hierarchies

Because of the variety of protocols used, resource hierarchies for WebLogic resources are constructed in a variety of different ways. [Table 7](#) outlines how WebLogic resource hierarchies for each different protocol must be constructed.

Table 7 Resource Hierarchies

Resource type	Protocol	Resource Hierarchy	Description
URL	HTTP	context path	Web application's context path. <i>Note:</i> For HTTPS resources use HTTP as the protocol, the WebLogic Enforcer plugin does not distinguish HTTP and HTTPS resources.
		resource_name1 resource_name2 ... resource_nameX	Each sub-resource is a step deeper into the directory hierarchy of the host machine, which may terminate at a specific file.
EJB	EJB	ejb_name	The name of the EJB resource, as defined in the <ejb-name> tag in the resource's deployment descriptor.
		method_name:param1: param2	The name of the EJB method. You can append optional parameters by separating them with a colon.
JNDI	JNDI	jndi_action	An action (lookup, modify, or list) performed on an external directory.
		resource_name1 resource_name2 ... resource_nameX	Each sub-resource is a step deeper into the directory hierarchy of the host machine, which may terminate at a specific resource.
JDBC	JDBC	jdbc_action	An action (reserve, admin, shrink, restore) performed on a database.
		resource_type	The type of persistent connection.
		resource_name	The name of the connection.

Table 7 Resource Hierarchies (cont'd)

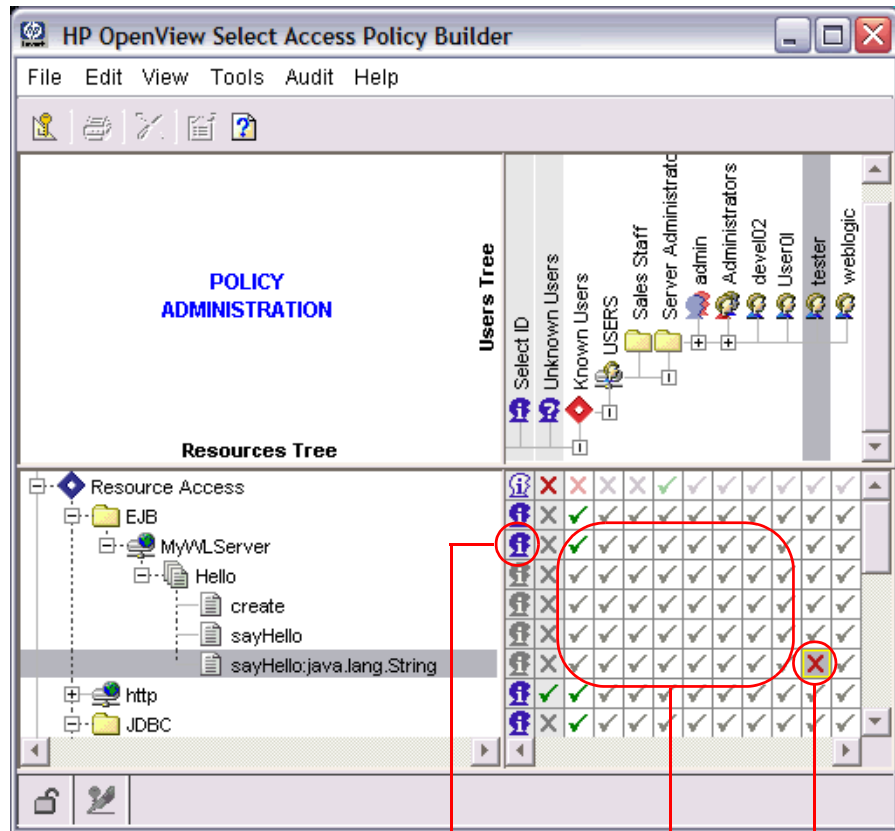
Resource type	Protocol	Resource Hierarchy	Description
JMS	JMS	jms_action	An action performed on a message. You can protect the send and receive functions.
		destination_type	The destination type (queue or topic) the message is being sent to or from.
		resource_name	The name of the queue or topic.
Web Service	webservice	context_path	The web application's context path.
		web_service_name	The web service name.
		method_name{param1 :param2...}	The name of the web service method. You can append optional parameters in a brace, separated by a colon.
Application	WebLogic	application_name	The name of the application.
Server	WebLogic	server_action	An action (boot, shutdown, lock, unlock) performed on the WebLogic Server.
		server_name	The name of the WebLogic Server. Note: You must define a boot action for a WebLogic Server and set the policy at the server_name level to define who can perform this action.
Administration	WebLogic	resource_name	The name of the administration resource.

An example EJB hierarchy

In the example shown in Figure 4, an administrator adds an EJB resource named `Hello`, as well as three sub-resources (methods):

- `create`
- `sayHello`
- `sayHello:java.lang.String`

All users, once authenticated, can access the EJB Hello, as well as the methods create and sayHello. Note that the user named “Tester” does not have permission to use the EJB method sayHello with the parameter java.lang.String.



Select Auth enabled and password authentication used. Unknown users are denied access until correct login credentials supplied.

Known users inherit access down the branch for the Hello resource.

“Tester” is allowed access to all EJB resources, except the EJB method & parameter.

Figure 4 Example Policy Shown with EJB Entries

To bypass protocols

The BEA WebLogic Enforcer sends requests for all services to the Validator, including JNDI lookups, and EJB, JDBC and JMS requests. This may cause a large number of Enforcer requests to be sent and significantly affect performance. You can specify services to ignore to prevent extra Validator requests.

- 1 Run the Policy Builder.
- 2 Click **Tools** → **Component Configuration**. The **Component Configuration** screen appears.
- 3 Select one of the following:
 - **Enforcer Plugins** to select all Enforcers.
 - **Generic Enforcer Plugin** to only select the WebLogic Enforcer.

- 4 Click **Edit** → **Properties**. The **Editing Enforcer Plugin Common Settings** screen appears.
- 5 Click the **Customizable Properties** tab.

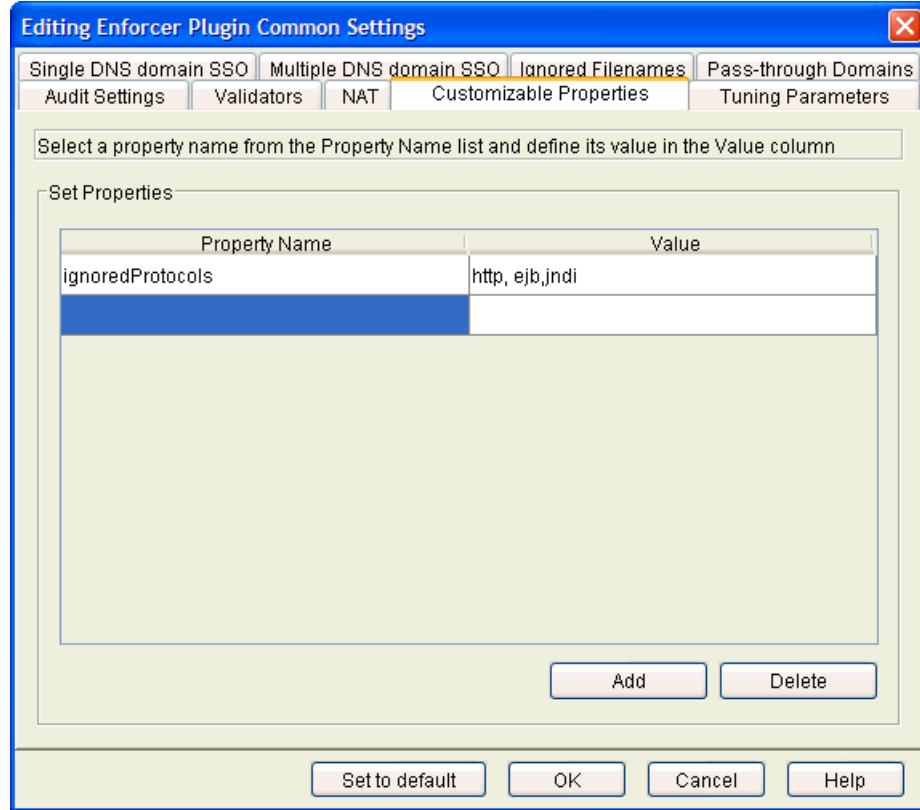


Figure 5 Enforcer Plugin Customizable Properties

- 6 Click **Add**.
- 7 Type `ignoredProtocols` in the **Property Name** field.
- 8 Type the protocols to be ignored in the **Value** field.
 - ▶ Enter the protocols in lower case letters. Separate protocols using a comma.
- 9 Click **OK**. A warning appears telling you to refresh the Enforcer plugin configuration for the changes to take effect immediately.
- 10 Click **OK** to implement the changes.

When to Use Default Security Providers

The WebLogic Server uses an internal LDAP user source to authenticate users who try to:

- Access critical system tools.
- Invoke operations on critical system administration Management Beans (MBeans).

Consequently, the WebLogic Server attempts to authenticate these users against its own data source, even after you integrate Select Access with it. To protect those resources, use the WebLogic Server's default security providers instead. This includes managing who can boot the server.

Setting Up Form-based Authentication

If you want to have your own custom login page instead of browser's login dialog box, you can setup Form-based authentication.

To setup Form-based authentication you define `auth-method` tag in you web application deployment descriptor (`web.xml`) file, and set its value to `FORM`. [Sample deployment descriptor for Form-based authentication](#) shows how to add this tag to your `web.xml` file.

Sample deployment descriptor for Form-based authentication

An example deployment descriptor is illustrated below:

```
<web-app>
...
  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>SAR realm</realm-name>
    <form-login-config>
      <form-login-page>/my_login.htm</form-login-page>
      <form-error-page>/my_fail_login.htm</form-error-page>
    </form-login-config></login-config>
  </login-config>
...
</web-app>
```

The `form-login-config` and `form-error-page` tags define the path and filename of your own login page and fail login page. For details on how to setup Form-based authentication in your web application, see [Chapter 2, Securing Web Applications](#), in the *BEA WebLogic Server Programming WebLogic Security Guide*.



Login pages should typically not include any protected resources. Otherwise, users may not be redirected back to the original page after a successful login. For details, see [Using Graphics on Login Pages](#) on page 33.

Using Graphics on Login Pages

If your login page has some image files in it, check that those image files are not protected by any Select Access access policies. You can use one of two methods:

- Setting up access in the Policy Builder: Set an Allow access policy for all images on the login page for all unknown users.
- Setting up Select Access' WebLogic Enforcer plugin to ignore all image files with the **Ignored Filenames** setup screen when creating your Enforcer configuration XML file. Make sure you check the **Ignore GIF images** and the **Ignore JPEG images** boxes. For details, see [Chapter 8, Configuring the Enforcer Plugins](#) in the *HP OpenView Select Access 6.2 Installation Guide*.

Setting Up Form-based Authentication with Multiple DNS Domain SSO

Multi-Domain SSO allows you to access content that is protected by the same Select Access system even when they exist in different DNS domains. For example, if your WebLogic Server is located on `server1.domain1.com`, and another web server is located on `server2.domain2.com`, you can setup Single Sign-on between them. To setup Form-based authentication with Multi-Domain SSO, you need to:

- Enable **Re-Login** on the WebLogic server.
- Define the Select Access login servlet as your web application's login page.
- Configure the Authentication server (from the Policy Builder) to use either:
 - The login form template provided by HP.
 - Your own custom login page.



Remember, your login pages should not include any protected resources. For details, see [Using Graphics on Login Pages](#) on page 33.

To set up SSO

- 1 Enable **Re-Login** on the WebLogic server using the following steps:
 - a Log in to the Admin console.
 - b Click **Lock & Edit**.
 - c Click the domain name on left panel (e.g. `myDomain`, `base_domain`, etc).
 - d Click **Web Applications** on the right panel.
 - e Select the **Relogin Enabled** check box.
 - f Click **Release Configuration**.
- 2 Configure each application to use Form-based authentication with its `web.xml` file so it uses Select Access' login servlet as the login page.
 - a Open the deployment descriptor `web.xml` file in a text editor.
 - b Add and/or configure the tags to listed in [Table 8](#) to this file.

- c Save the changes to this file.

Table 8 web.xml Tags Required

Tag	Sub-tag	Description
servlet	servlet-name	Sets name to any string. This string must meet the application's requirements.
	servlet-class	Sets the class of the login servlet. The class must be fully-qualified.
	run-as	<p>Sets the role-name to a WebLogic administrator (with the corresponding role) defined by the default Authentication Provider. The administration role allows Select Access' servlet to perform Perimeter Authentication.</p> <p>Tip: When you created Select Access' security realm, all default user, group, role and policies are added to it. This includes a default user that Select Access uses to boot the WebLogic Server. You can use this name as the run-as role name.</p>
servlet-mapping	servlet-name	<p>This string must:</p> <ul style="list-style-type: none"> • Meet the application's requirements. • Must match the <code>servlet-name</code> in the <code>servlet</code> tag.
	url-pattern	Maps the servlet to a url string pattern. This string must meet the application's requirements.
login-config	auth-method	Sets the method of authentication. Define this method as FORM.
	realm-name	Sets a specific realm. If you do not supply a realm name, the default realm is used.
	form-login-config	<p>Sets:</p> <ul style="list-style-type: none"> • <code>form-login-page</code> to the same URL pattern as in <code>servlet-mapping</code>. • <code>form-error-page</code> to the application's login error page. Leave the value of this tag empty; Select Access does not use this value.

See [Sample web.xml file](#) for an example of a completed `web.xml` file.

- 3 Configure an Authentication Server and corresponding login form in the Policy Builder. For details, see [Chapter 6, Setting Up Authentication Services](#) in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

- 4 Run the Setup Tool and configure SSO on the Enforcer plugin for each third-party web server and WebLogic Server. For details on how to configure SSO for Enforcer plugin, see [Chapter 8, Configuring the Enforcer plugins](#), in the *HP OpenView Select Access 6.2 Installation Guide*.

Sample web.xml file

The following is a complete sample of a server configuration file.

```
<web-app>
  <!-- Define the Servlets within the Web Application -->
  <servlet>
    <servlet-name>LoginServlet</servlet-name>

    <servlet-class>com.hp.ov.selectaccess.solutions.weblogic.securityprovide
r.Login.LoginServlet</servlet-class>
    <run-as>
      <role-name>weblogic</role-name>
    </run-as>
  </servlet>
  <servlet>
    <servlet-name>PersonalizationServlet</servlet-name>
    <servlet-class>PersonalizationServlet</servlet-class>
  </servlet>
  <!-- Define Servlet mappings to urls -->
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>login.htm</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>PersonalizationServlet</servlet-name>
    <url-pattern>PersonalizationServlet</url-pattern>
  </servlet-mapping>
  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>SAR realm</realm-name>
    <form-login-config>
      <form-login-page>/login.htm</form-login-page>
      <form-error-page></form-error-page>
    </form-login-config>
  </login-config>
</web-app>
```

Setting Up Perimeter Authentication

Perimeter authentication is best used when:

- You do not plan to let users log in from the WebLogic Server.

- Due to a limitation of WebLogic Server's web container, users who have set up Perimeter Authentication cannot use Forms or HTTP Basic Authentication. Therefore you want users signing in from other systems protected by Select Access to also log into the WebLogic Server.

By default, when sessions time out (that is, the token has expired), the WebLogic Server shows users the unauthorized error page (HTTP 401 error page). Note that Select Access' WebLogic Enforcer plugin cannot redirect a user to login server. Instead, you can setup your own 401 error page to replace WebLogic default one. For details, see [To modify your HTTP 401 error page](#) on page 38.

To set up Perimeter authentication

- 1 Configure the Select Access Identity Assertion Provider as described in [Table 5](#).
- 2 Modify the web application's deployment descriptor to set authentication method to `CLIENT-CERT`. Do this by:
 - Adding an `auth-method` tag to your `web.xml` file.
 - Set its value to `CLIENT-CERT`.

[Sample web.xml file](#) shows how to add this tag to your `web.xml` file.

Sample web.xml file

The following sample shows how to add the `auth-method` tag.

```
<web-app>
...
  <login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>SAR realm</realm-name>
  </login-config>
...
</web-app>
```

To modify your HTTP 401 error page

- 1 Create your own 401 error page and then do one of the following:
 - Add a link to the login server.
 - Have the page redirect the user directly to the login server.
- 2 Define custom error pages in the `error-page` tag of the web application's deployment descriptor (`web.xml`).

For more information about setting up custom error page, see [Customizing HTTP Error Responses](#) in [Chapter 3, Configuring Web Application Components](#) in *Developing Web Applications for WebLogic Server*.

Setting Up Authentication Via a Proxy Server

This method is ideally suited to organizations that:

- Want the Proxy server to authenticate users.
- Want full protection to all the resources support by Select Access' WebLogic Enforcer plugin, not only the URL resources protected by Proxy server.

To set up authentication via a proxy

- 1 Set up your WebLogic Server to use a web server as a proxy. For details, see BEA's *Using Web Server Plug-Ins with WebLogic Server* guide.
- 2 Install the corresponding Enforcer plugin on the web server proxy.
- 3 In the Policy Builder, set up your required authentication methods for the web server proxy.
- 4 Install and configure Select Access' WebLogic Enforcer plugin as described in [Setting Up Perimeter Authentication](#) on page 36.
- 5 In the Policy Builder, set access policies for all WebLogic resources for the WebLogic service entry. You do not need to set access policies for URL resources for the web service that is your proxy, except if the URLs are for the static HTML pages on your web server.

Setting Up Certificate Authentication

To use Certificate Authentication, set up your environment as described in [Setting Up Form-based Authentication with Multiple DNS Domain SSO](#) on page 34. However, there are a few differences to take note of:

- You do not need to choose a login form; Certificate-based authentication uses digital certificates and not forms.
- Before you can setup the Certificate Authentication Server in the Policy Builder, load the root certificate into the Select Access directory server. For details on how to configure a Certificate Authentication server, see [Certificate Authentication Service](#) on page 115 the *HP OpenView Select Access 6.2 Policy Builder Guide*.
- Set up your WebLogic Server to use two-way SSL and load the root certificate to WebLogic Server's trust keystore. For details see [Chapter 7, Configuring SSL](#) on page 7-1 in the *BEA WebLogic Server Managing WebLogic Security Release 8.1*.
- Based on your security requirements, you may *optionally* choose to store user certificates in the Select Access directory server. Only if you want to compare each user's certificate with the one stored in Select Access, then you *must*:
 - Upload the user's certificate to the corresponding user's LDAP entry,
 - Check the **Require certificates to be stored on the directory server** box in the **New Certificate Server** dialog box.

Setting Up Multiple Authentication Methods

You can setup multiple authentication methods by adding more than one authentication server to SelectID Properties dialog box. In addition to configuring multiple authentication methods, you also need to use a special login form, `multiauth_form.html`. This form contains multiple authenticatin fields that correspond the credential collection for specific authentication methods. All authentication form elements are supported by Select Access' WebLogic Enforcer plugin, except the registration element.

If you use Certificate authentication with other authentication methods, Select Access' WebLogic Enforcer plugin tries certificate authentication first:

- If the user successfully authenticated via his/her certificate, then authentication is considered successful. The Enforcer plugin does not need to display the login form.
- If the user does not successfully authenticate, the Enforcer plugin displays the login form to let user login via other authentication methods.

Setting Up Personalization

Select Access' WebLogic Enforcer plugin supports personalization for JSPs, servlets, and EJBs. With information either obtained from the directory server, or provided in real-time by a registering user, the communications between the WebLogic Server and user is altered to fit that user's stated needs as well as needs perceived by the business based on the available user information.

Unlike Select Access' web server-based Enforcer plugins, the WebLogic Enforcer plugin neither adds the prefix `HTTP_SA` to a personalization variable name, nor does it change the variable name to uppercase. The variable name is exactly the same as you defined in the Policy Builder.

For more details about personalization, see [Chapter 6, Authentication fundamentals: Select ID and personalization](#), in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

How it Works in Select Access' WebLogic Enforcer Plugin

Personalization data is stored as key/value pairs in the `SAP13n` object, which is part of the Select Access Principal object. WebLogic applications can use this data by giving the corresponding key name to the `SAP13n` object.

To extract personalization data

- 1 Call the corresponding method:
 - For JSPs and servlets: `getUserPrincipal()` of the request object.
 - For EJB: `getCallerPrincipal()` of the EJB context.
- 2 To extract personalization data, call `getP13n()` by using the Principal object obtained in step [Step 1](#).

[Extracting personalization data from a servlet](#) shows how to extract personalization data from a servlet. In this example, the `PersonalizationServlet` displays all personalization data retrieved from Principal in a table.

Extracting personalization data from a servlet

The following sample shows how personalization data is extracted.

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.security.Principal;
```



```

import
com.hp.ov.selectaccess.solutions.weblogic.securityprovider.principal.SAP
rincipalImpl;
import
com.hp.ov.selectaccess.solutions.weblogic.securityprovider.principal.SAP
13n;

/**
 * This is an example to show how a servlet retrieves personalization
 data.
 */
public class PersonalizationServlet extends HttpServlet {

    public void doGet (HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
        SAPPrincipalImpl saPrincipal = null;
        SAP13n p13n = null;

        // Get principal from http request
        Principal principal = request.getUserPrincipal();
        if (principal != null && principal instanceof SAPPrincipalImpl) {
            saPrincipal = (SAPPrincipalImpl)principal;

            // Get personalization object from principal
            p13n = saPrincipal.getP13n();
        }

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html>" +
"<head><title>Select Access WebLogic Enforcer plugPersonalization
Test</title></head>");

        if (p13n != null && !p13n.isEmpty()) {

            out.println("<H3>Personalization info in principal:</H3>");
            out.println("<table border=1><tr><td><b>Key</b>
</td><td><b>Value</b></td></tr>");

            // Get a collection of keys of all personalization attribute
Set keySet = p13n.keySet();

            Iterator iterator = keySet.iterator();

            while (iterator.hasNext()) {
                String key = (String)iterator.next();
                String value = null;

```

```

        // Get peronalization attribute by key
        value = (String)p13n.getAttribute(key);
        out.println("<tr><td>" + key + "</td><td>" + value +
            "</td></tr>");
    }

    out.println("</table>");

    } else {
        out.println("<p>Personaliztion info not found in
            principal.</p>");
    }

    out.println("</body></html>");
    out.close();
}
}

```

Integration with the WebLogic Portal

The Select Access' WebLogic Enforcer plugin can be integrated with WebLogic Portal Server versions 8.1 SP3 or later.

Since WebLogic Portal Server has its own proprietary authorization management mechanism, which is not accessible to external security providers, the Select Access' WebLogic Enforcer plugin will only provide authentication support for WebLogic Portal Server. Thus, access control for portal resources (such as desktop, book, page, portlet, etc) must be managed in the WebLogic Administration Portal. Security rules for other resources will still be managed in Select Access Policy Builder.

In Select Access, multiple groups can have the same name (cn) as long as their DN's are different. However, the WebLogic Portal Server doesn't allow duplicate group names, so you must make sure there are no duplicated group names defined in Select Access. In Select Access Policy Builder, folders are used to organize users and groups. Although folders are similar to groups in some aspects, the Enforcer plugin doesn't export folder names to the WebLogic Portal, so folders won't appear in the WebLogic Administration Portal.

To integrate Select Access with the WebLogic Portal Server, you simply follow the instructions to integrate the Select Access' WebLogic Enforcer plugin with WebLogic Server first, and then configure the Enforcer plugin to work with the WebLogic Portal Server by following the steps in [Table 9](#).

Table 9 Configuring Select Access' WebLogic Enforcer Plugin for WebLogic Portal

Step	Details
<p>Step 1: Configure the Enforcer plugin with your Select Access directory server connection information. This information provides user and group information to the WebLogic Portal, the Enforcer plugin needs</p> <p>Note: This solution currently allows for Portal users on a single directory only.</p>	<p>Open the Enforcer configuration file <code>bea_enforcer.properties</code>, and add the following configuration items:</p> <ul style="list-style-type: none"> • DirectoryServer: The URL of directory server, including port number. For example: <code>DirectoryServer=ldap://localhost:389</code> • DirectoryServerPrincipal: The username for login to directory server. If empty or not defined, the Enforcer plugin will try to connect to the directory server as an anonymous user. • DirectoryServerCredential: The login credentials necessary to access the directory server. • UserStoreLocation: The Select Access user store location. For example: <code>UserStoreLocation=ou=people,dc=mydomain,dc=mycompany,dc=com</code> • UserIDAttribute: The attribute of user entry in LDAP server to uniquely identify a user in WLS. For example: <code>UserIDAttribute=uid</code> <p>Note: To make Select Access' WebLogic Enforcer plugin work well with the WebLogic Portal, you have to define the personalization data <code>UID</code> in Policy Builder. You also need to make sure the value of <code>UserIDAttribute</code> is same as the value of <code>UID</code>. For details about setting personalization data <code>UID</code>, see page 37, Table 7, Step 3.</p> <p>GroupNameAttribute: The attribute of group entry in LDAP server to uniquely identify a group in WLS. For example: <code>GroupNameAttribute=cn</code></p>

Table 9 Configuring Select Access' WebLogic Enforcer Plugin for WebLogic Portal (cont'd)

Step	Details
<p>Step 2: Configure Security Rules in Policy Builder.</p>	<p>When the WebLogic Portal Server starts up, the Portal Framework needs to access some resources anonymously. You need to setup proper security policy to allow anonymous access to those resources. Some common resources needed by all portal servers include:</p> <ul style="list-style-type: none"> • JNDI Resources: <ul style="list-style-type: none"> lookup/<portal_app_name> lookup/weblogic lookup/portalFrameworkPool lookup/jws lookup/cgDataSource • JMS Resources: <ul style="list-style-type: none"> receive/queue/cgJWSQueue • JDBC Resources: <ul style="list-style-type: none"> reserve/ConnectionPool/cgPool <p>Other resources may also require to be accessed anonymously during portal server startup depending on the implementation of your portal application. To configure those resources, see Configure Resources Required During Server Startup on page 44.</p>
<p>Step 3: Reboot WebLogic Portal Server to implement these changes.</p>	<p>N/A</p>
<p>Step 4: Configure Security Rules in WebLogic Administration Portal to browse users and groups provided by Select Access Authenticator.</p>	<ol style="list-style-type: none"> 1 Login to WebLogic Administration Portal. 2 In the Service Administration page, click Authentication Hierarchy Service from the left pane. 3 Enter <code>SAAuthenticator</code> in the textbox next to Provider to Add to Build List. 4 Click Update & Build Tree. 5 In the Users and Groups page, select <code>SAAuthenticator</code> from the Browse User-Groups from drop-down list. 6 Verify that the Select Access users and groups appear in the screen.

Configure Resources Required During Server Startup

If your Portal application cannot startup correctly after installing Select Access' WebLogic Enforcer plugin, it is probably because the anonymous access to some resources were denied by the Select Access Policy Validator.

To determine resource denials

- 1 Modify the Enforcer plugin's log level to `info`.
- 2 Restart your server.
- 3 Check the Enforcer plugin's audit output to determine which resources were denied by Policy Validator.
- 4 Choose one of the following approaches to solve the problem:
 - Setup a security policy to allow anonymous access to the resources.
 - **(Recommended)** Grant the caller a proper `run-as` role in its deployment descriptor. You can only choose this option if the caller is a component of your application instead of a component of Portal Framework.

Configure Security Policy and User Profiles in the WebLogic Administration Portal

Once you are able to see Select Access users and groups in the WebLogic Administration Portal, you can:

- Create visitor entitlement based on Select Access users and groups.
- Build security policy by using that entitlement
- Edit user's profiles. Please refer to WebLogic Portal documents for details about management security and user profile in the Portal.

The Select Access' WebLogic Enforcer plugin only allows read-only access to the Select Access user store. Editing the user, group or group membership information with the WebLogic Administration Portal won't have any effect.

